

REMARKS

Claims 1-51 are pending, with claims 1, 12, 29, 31, 33, 36, 42, 46, 48 and 50 being in independent form.

Claims 29, 45, 46, and 50 have been amended to remedy informalities. No new matter has been added.

An Appendix with four (4) exhibits is included with this Amendment.

Reconsideration and allowance of the above-referenced application are respectfully requested in light of the following remarks.

Rejections under 35 U.S.C. §112 First Paragraph (Best Mode)

Claims 1-51 stand rejected under 35 U.S.C. §112, first paragraph, because the best mode contemplated by the inventor has allegedly not been disclosed. The rejection is respectfully traversed.

The Examiner cites passages and phrases in the Specification, such as “in various embodiments” and “the order of description should not be construed as to imply that these operations are necessarily order dependent,” as alleged evidence of concealment of the best mode and as examples of language in the Specification that raised questions in the Examiner’s mind as to whether the best mode has been set forth. Applicant respectfully asserts that such statements in Applicant’s specification are not evidence of concealment.

The best mode inquiry involves two parts. First, a determination is made as to whether, at the time of the application, the inventor knew of a mode of practicing the claimed invention that the inventor considered to be better than any other. Second, if it is determined that the inventor knew of a mode that he considered better than any other, a determination is made as to whether the disclosure is adequate to enable one skilled in the art to practice the best mode. *See* MPEP 2165.03; *Chemcast Crop. v. Arco Industries Corp.*, 913 F.2d 923 (Fed. Cir. 1990); *Eli Lilly & Co. v. Barr Laboratories, Inc.*, 251 F.3d 955 (Fed. Cir. 2001). Satisfaction of the best mode requirement is a question of fact subject to the clearly erroneous standard, but the absence

of evidence to support the basic tenets of a best mode violation satisfies that standard. *See* *Bruning v. Hirose*, 161 F.3d 681, 687 (Fed. Cir. 1998). Because the best mode inquiry often requires information that is unavailable to examiners during *ex parte* examination, examiners are instructed to assume that the best mode is disclosed unless evidence is presented that is inconsistent with that assumption. *See* MPEP 2165.03.

Firstly, the examiner's evidence does not show that at the time of the application, the inventor knew of a mode of practicing the claimed invention that the inventor considered to be better than any other. The Applicant respectfully asserts that the cited language from the specification merely suggests the possibility of other embodiments beyond those explicitly disclosed in the specification. Such language is not inconsistent with a proper disclosure of the best mode in the specification. Secondly, assuming *arguendo* that the asserted evidence satisfies the first prong of the best mode inquiry (which it does not), the evidence does not show that the disclosure is inadequate to enable one skilled in the art to practice the best mode. The Examiner has not provided any evidence that shows that the quality of the disclosure in the instant application is so poor as to effectively result in concealment. *See* MPEP 2165.04. Moreover, one cannot conclude that there is a concealment of the best mode merely from a possibility that there may be other embodiments beyond those explicitly disclosed in the specification. Accordingly, the Examiner has not met the requisite legal burden for showing the lack of best mode.

Rejections under 35 U.S.C. §112 First Paragraph (Enablement)

Claims 1, 11, 21, 23, 25, 30-33, 36, and 47-50 stand rejected under 35 U.S.C. §112, first paragraph for allegedly lacking enablement. The rejection is respectfully traversed.

Any analysis of whether a particular claim is supported by the disclosure in an application requires a determination of whether that disclosure, when filed, contained sufficient information regarding the subject matter of the claims as to enable one skilled in the pertinent art to make and use the claimed invention. *See* MPEP 2164.01. The standard for determining whether the specification meets the enablement requirement was cast in the Supreme Court

decision of *Mineral Separation v. Hyde*, 242 U.S. 261, 270 (1916) which postured the question: is the experimentation needed to practice the invention undue or unreasonable? That standard is still the one to be applied. *In re Wands*, 858 F.2d 731, 737, 8 USPQ2d 1400, 1404 (Fed. Cir. 1988).

In order to make a rejection, the examiner has the initial burden to establish a reasonable basis to question the enablement provided for the claimed invention. According to *In re Wright*, 999 F.2d 1557, 1562, 27 USPQ2d 1510, 1513 (Fed. Cir. 1993), the examiner must provide a reasonable explanation as to why the scope of protection provided by a claim is not adequately enabled by the disclosure. A specification which contains a teaching of the manner and process of making and using the invention in terms which correspond in scope to those used in describing and defining the subject matter sought to be patented must be taken as in compliance with the enablement requirement unless there is a reason to doubt the objective truth of the statements contained therein which must be relied on for enablement. *See id.*; *In re Marzocchi*, 439 F.2d 220, 223-224 (CCPA 1971); *In re Wright*, 999 F.2d at 1561-62.

Claims 1 and 36. Claims 1 and 36 recite the feature of a “generation technique.” To the extent that the Examiner’s argument is intelligible, it appears that the Examiner has alleged that this feature is unclear or undefined in the specification. As a preliminary matter, the Examiner has not provided a shred of evidence supporting the position that this feature is “unclear” or “undefined.” In fact, the very evidence that the Examiner proffers shows just the opposite. The specification describes a “generation technique” and provides examples of such, namely a triangular technique and a quadrilateral technique. *See* Specification, p. 9 lines 13-29. The specification also describes how to perform the given examples of generation techniques. *See* Specification, p. 10 line 7 – p. 14 line 18.

Secondly, the Examiner provides no evidence that there is a reason to doubt the objective truth of the statements in the specification. One of ordinary skill in the art who reads the specification will understand what Applicant’s claimed “generation technique” is and how to make and use such. Thus, the Examiner has not met the legal burden for showing a lack of enablement for this feature.

Claims 1, 25, 32, 33, 36, 49, and 50. Claims 1, 25, 32, 33, 36, 49, and 50 recite the features of “selecting,” “applying,” or “determining” within a computing environment. It appears that the Examiner has alleged that these features are unclear or undefined in the specification. The Examiner has not provided evidence supporting the position that this feature is “unclear” or “undefined.” Furthermore, the specification describes examples of the selecting, applying, and determining operations. *See, e.g.*, Specification, p. 10, lines 1 – 5; p. 16 lines 9 – 12. The Examiner provides no evidence that there is a reason to doubt the objective truth of the statements in the specification. Thus, the Examiner has not met the legal burden for showing a lack of enablement for this feature.

Furthermore, these features were well known to those of ordinary skill in the art as of the filing date. A specification need not teach, and preferably omits, what is well known in the art. *See* MPEP 2164.01. The Examiner has not alleged that such operations are not within the skills of a person having ordinary skill in the art. One of ordinary skill in the art who reads the specification will understand how to make and use the selecting, applying, and determining features. Because such operations are well known in the art, further description in the specification is not necessary for purposes of enablement.

Claims 11 and 23. Claims 11 and 23 recite the feature of a “non-regularized” Boolean operation. It appears that the Examiner has alleged that this feature is unclear or undefined in the specification. The Examiner has not provided evidence supporting the position that this feature is “unclear” or “undefined.”

Furthermore, “non-regularized” Boolean operations were well-known in the art as of the filing date. *See, e.g.*, Rossignac, “Issues on Features-Based Editing and Interrogation of Solid Models,” *Comput. & Graphics*, Vol. 14, No. 2, pp. 149-172 (1990) (Exhibit A); Rossignac and Requicha, “Constructive Non-Regularized Geometry,” *Computer-Aided Design*, Vol. 23, No. 1, pp. 21-32 (1991) (Exhibit B). These references describe non-regularized Boolean operations well before the filing date of the instant application. A specification need not teach, and preferably omits, what is well known in the art. *See* MPEP 2164.01. Because “non-regularized” Boolean operations are well known in the art, further description in the specification is not

necessary for purposes of enablement. Thus, the Examiner has not met the legal burden for showing a lack of enablement for this feature.

Claims 21, 30, 31, 47, and 48. Claims 21, 30, 31, 47, and 48 recite the feature of a “wire body.” The Examiner asserts that the term “wire body” is not disclosed in the specification. It appears that the Examiner has alleged that this feature is unclear or undefined in the specification. The Examiner has not provided evidence supporting the position that this feature is “unclear” or “undefined.” Also, an enablement rejection that is made merely because a term is recited in the claims but not in the specification is improper. The claimed limitation in and of itself may enable one of ordinary skill in the art to make and use the claim containing the limitation. *See* MPEP 2164. The Examiner has not alleged that the term “wire body,” as recited in the claims, is not enabling in and of itself. Thus, the Examiner has not met the legal burden for showing a lack of enablement for this feature.

Furthermore, Applicant submits that “wire body” was well-known in the art as of the filing date. *See* Lambda Research Corp., *Illuminations* newsletter, vol. 2, issue 1, Jan. 2000, pp. 1-4 (Exhibit C); Schild *et al.*, “Open Data Exchange with HP PE/SolidDesigner,” *Hewlett-Packard Journal*, October 1995, pp. 35-50 (Exhibit D). The references describe a wire body as synonymous with a wire frame. Because “wire body” is well known in the art, further description in the specification is not necessary for purposes of enablement.

Claims 25, 32, 33, 49, and 50. Claims 25, 32, 33, 49, and 50 recite in part, “valid ones of said faces.” The Examiner asserts that the term “valid” is unclear because the Specification discusses “invalid” faces but not “valid” faces. The Examiner has not provided evidence supporting the position that this feature is “unclear” or “undefined.” Also, an enablement rejection that is made merely because a term is recited in the claims but not in the specification is improper. The claimed limitation in and of itself may enable one of ordinary skill in the art to make and use the claim containing the limitation. *See* MPEP 2164. The Examiner has not alleged that the term “valid,” as recited in the claims, is not enabling in and of itself. Thus, the Examiner has not met the legal burden for showing a lack of enablement for this feature.

Furthermore, Applicant submits that the term “valid” is sufficiently clear in the specification in the specification. One of ordinary skill in the art will reasonably understand that “valid” is the opposite of “invalid.” The specification describes various factors for determining whether a face is invalid. *See* Specification, p. 16, lines 13-21. One of ordinary skill in the art will reasonably understand that a “valid” face is a face that is determined as not invalid, based on the factors described in the specification. Thus, the term “valid” is sufficiently clear.

Rejection under 35 U.S.C. §112 second paragraph

Claims 1 and 36 stand rejected under 35 U.S.C. §112, second paragraph for allegedly being indefinite. The rejection is respectfully traversed.

The Examiner alleges that the feature “generation technique” in claims 1 and 36 is unclear and that it is difficult to determine a definition for this term because the Specification read “[i]n various embodiments, the generation techniques being considered may include, but are not limited to” Applicant submits that the definition of “generation technique” is definite in light of the specification. The Specification describes a “generation technique to generate a data representation to model a fillet weld bead,” provides examples of generation techniques, and describes how to perform such. *See* Specification, p. 9 line 13 – p. 14 line 18. In light of the Specification, and as addressed above, one of ordinary skill in the art will reasonably discern that “generation technique” refers to a technique used to generate a data representation to model geometries, such as a data representation to model a fillet weld bead. Accordingly, this rejection should be withdrawn.

Rejections under 35 U.S.C. §103(a)

Rejection over Subrahmanyam in view of CAI

Claims 1-4, 7-11, 12, 15-39, 42, and 45-51 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Subrahmanyam *et al.* (erroneously written as “Subrashekar” in the Office Action), “Feature Attributes and Their Role in Product Modeling,” Solid Modeling

'95, Salt Lake City, Utah, (1995) (hereafter referred to as "Subrahmanyam"), and further in view of Computer & Automation Institute, "PROARC, No. 7831, CAD-Based Programming System for Arc Welding Robots in One-Off Production Runs," ESPRIT, (Jan. 16, 2001) (hereafter referred to as "CAI"). The rejection is respectfully traversed.

Claim 1. Claim 1 recites in part, "selecting ... a generation technique based at least in part on the result of said examining; and applying ... the selected generation technique to generate a data representation of the fillet weld bead."

The Examiner asserted that Subrahmanyam teaches a method to generate a solid model that includes selecting a generation technique based at least in part on the result of said examining, and applying the selected generation technique to generate a data representation of the solid model. For this proposition, the Examiner relies on sections 4.2 - 4.4 and Table 1 of Subrahmanyam. In the cited portions of Subrahmanyam, operations are performed on a solid model and then attributes are processed after the operation. How an attribute is processed may be based on the operation performed and the attribute in question. For example, Subrahmanyam shows that a face normal vector attribute may or may not change, depending on the operation. However, there is no teaching or suggestion that a data representation of a shape is generated by processing attributes. Subrahmanyam merely shows how the attributes of a shape may be affected by operations on the shape. On the other hand, claim 1 recites selecting a generation technique and applying the generation technique to generate a data representation of a fillet weld bead. Therefore, Subrahmanyam does not teach or suggest this feature of claim 1. Accordingly, claim 1 is in condition for allowance for at least this reason.

Claims 2-4, 7-11. Claims 2-4 and 7-11 are dependent from claim 1 and are allowable for at least the reasons set forth above.

Claim 12. Claim 12 recites in part, "generating ... a tool based at least in part on the constructed profile." The Examiner asserts that Subrahmanyam teaches "generating with the computer environment, a tool based at least in part on a profile," citing p. 117, col. 2, line 5 and Figure 4 (erroneously written as Figure 5 in the Office Action) of Subrahmanyam. The cited portion of Subrahmanyam shows a starting block with attributes, a slot with attributes, and a hole

with attributes. There is no teaching or suggestion in the cited portion of Subrahmanyam of generating a tool based on a constructed profile. Assuming for the sake of argument that attributes are profiles, the cited portion of Subrahmanyam shows that the starting block, the slot, and the hole have attributes; there is no teaching that they are generated based on the attributes.

The cited portion of Subrahmanyam also shows subtracting the slot and the hole from the starting block. The Examiner relies on Subrahmanyam's showing of subtracting of the slot and the hole from the starting block as teaching generating a tool and conditionally trimming the tool. It appears to Applicant that the Examiner is treating the starting block as the "tool," because the Examiner is citing the subtracting as teaching trimming the tool and the object being "trimmed" in Subrahmanyam is the starting block. However, there is no explicit showing in Subrahmanyam of how the starting block is generated. Furthermore, as describe above, Subrahmanyam merely shows that the starting block has attributes. It does not teach or suggest generating the starting block based on attributes. Thus, Subrahmanyam also does not disclose "generating ... a tool based at least in part on a constructed profile." Accordingly, claim 12 is in condition for allowance.

Claims 15-28. Claims 15-28 are dependent from claim 12 and are allowable for at least the reasons set forth above.

Claim 29. Claim 29 recites in part, "locating ... one or more bodies referred to by a plurality of faces of a plurality of components." The Examiner asserts that Subrahmanyam teaches locating one or more bodies referred to by the faces. However, the cited portions of Subrahmanyam (Subrahmanyam, Section 3.7 lines 1-6, 10-11; Figure 4) make no mention of locating bodies referred to by faces. Subrahmanyam Section 3.7 shows attributes representing relationships between entities and applying a parallel attribute to a pair of faces, but not locating bodies referred to by faces. An attribute is defined in Subrahmanyam as "a characteristic quality or property which associates meaning to an entity, significant to a particular stage in the life cycle of a product," and gives as examples of attributes the color of a face, type of thread, type of feature, or relationships between two faces. *See* Subrahmanyam, Section 1 (Introduction). This definition and the given examples suggest that attributes are not the same as bodies. Nor is the

locating of bodies apparent from Figure 4 of Subrahmanyam. Figure 4 of Subrahmanyam shows a design view that includes a starting block with attributes, a slot with attributes, and a hole with attributes. There is no teaching in Subrahmanyam Figure 4 that the starting block, the slot, and the hole are located or that they are referred to by faces. Thus, the Examiner has not shown that Subrahmanyam teaches or suggests "locating ... one or more bodies referred to by a plurality of faces of a plurality of components." Accordingly, claim 29 is in a condition for allowance.

Claim 30. Claim 30 is dependent from claim 29 and is allowable for at least the reasons set forth above.

Claim 31. Claim 31 recites in part, "conditionally forming ... a wire body, if, data representations of more than one edge are replicated." The Examiner asserted that Subrahmanyam teaches conditionally forming a wire body if data representations of more than one edge are replicated (Subrahmanyam, Section 4.4, lines 12-17; Figure 6(b)). However, the cited portion of Subrahmanyam does not teach conditionally forming a wire body. The cited portions of Subrahmanyam show copying attributes of faces. Copying attributes of faces does not form a wire body in and of itself. Nor is it apparent that the shape depicted in Figure 6(b) of Subrahmanyam is a wire body. Thus, the cited portions of Subrahmanyam does not teach or suggest forming a wire body. Accordingly, claim 31 is in a condition for allowance.

Claim 32. Claim 32 is dependent from claim 31 and is allowable for at least the reasons set forth above.

Claim 33. Claim 33 recites in part, "copying and extending ... the selected valid ones of said faces into bodies." The Examiner asserts that Subrahmanyam teaches copying and extending, within the computing environment, the selected valid ones of said faces into bodies. However, the cited portions of Subrahmanyam (Section 4.4, lines 12-15; page 121, "Merging Operations," ¶¶ 1-4) do not teach or suggest extending a face into a body. Subrahmanyam shows handling of attributes, such as copying or deleting attributes, between faces after a split or a merge. Such handling of attributes does not extend a face into a body. A face is not extended into a body merely because its attributes may have been changed by the copying or deleting of

attributes. Thus, Subrahmanyam does not teach “copying and extending ... the selected valid ones of said faces into bodies.” Accordingly, claim 33 is in a condition for allowance.

Claim 34-35. Claims 34-35 are dependent from claim 33 and are allowable for at least the reasons set forth above.

Claim 36. Claim 36 is directed to an apparatus having a storage medium. The storage medium has stored thereon programming instructions designed to enable the apparatus to, in part, “select a generation technique based at least in part on the result of said examination; and applying the selected generation technique to generate a data representation of a fillet weld bead of the fillet welding operation.” As set forth above with respect to claim 1, Subrahmanyam has no teaching or suggestion of selecting a generation technique and applying the selected technique to generate the shape. Accordingly, claim 36 is in a condition for allowance.

Claims 37-39. Claims 37-39 are dependent from claim 36 and are allowable for at least the reasons set forth above.

Claim 42. Claim 42 is directed to an apparatus having a storage medium. The storage medium has stored thereon programming instructions designed to enable the apparatus to, in part, “generate a tool based at least in part on the constructed profile.” As set forth above with respect to claim 12, Subrahmanyam does not teach or suggest generating a tool based at least in part on the constructed profile. Accordingly, claim 42 is in a condition for allowance.

Claim 45. Claim 45 is dependent from claim 42 and is allowable for at least the reasons set forth above.

Claim 46. Claim 46 is directed to an apparatus having a storage medium. The storage medium has stored thereon programming instructions designed to enable the apparatus to, in part, “locate one or more bodies referred to by a plurality of faces of a plurality of components.” As set forth above with respect to claim 29, Subrahmanyam does not teach or suggest locating bodies referred to by faces. Accordingly, claim 46 is in a condition for allowance.

Claim 47. Claim 47 is dependent from claim 46 and is allowable for at least the reasons set forth above.

Claim 48. Claim 48 is directed to an apparatus having a storage medium. The storage medium has stored thereon programming instructions designed to enable the apparatus to, in part, “conditionally form a wire body, if, data representations of more than one edge are replicated.” As set forth above with respect to claim 31, Subrahmanyam does not teach or suggest conditionally forming a wire body. Accordingly, claim 48 is in a condition for allowance.

Claim 49. Claim 49 is dependent from claim 48 and is allowable for at least the reasons set forth above.

Claim 50. Claim 48 is directed to an apparatus having a storage medium. The storage medium has stored thereon programming instructions designed to enable the apparatus to, in part, “copy and extend the selected valid ones of said faces into bodies.” As set forth above with respect to claim 33, Subrahmanyam does not teach or suggest extending faces into bodies. Accordingly, claim 50 is in a condition for allowance.

Claim 51. Claim 51 is dependent from claim 50 and is allowable for at least the reasons set forth above.

Rejection over Subrahmanyam in view of CAI and Wang

Claims 5, 6, 13, 14, 40, 41, 43 and 44 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Subrahmanyam and CAI as applied to claims, 1, 12, 36 and 42 above, and Wang, et al., “The Design and Fabrication of Welded Tubular Joints Using Solid Modeling Techniques,” 2nd ACM Solid Modeling, 1993, hereafter referred to as “Wang.” The rejections are respectfully traversed.

Claims 5 and 6. Claims 5 and 6 are dependent from claim 1, and are in a condition for allowance for at least the reasons set forth above. Additionally, the cited portions of Wang do not teach or suggest a triangular or quadrilateral profile. Wang shows modeling of tubular joint welds. There is no teaching or suggestion in the cited portions of Wang (p. 430, last paragraph of Introduction; Section 2.4; Figure 8b) of a triangular or quadrilateral profile. There is no teaching or suggestion that a profile is of a particular shape. Furthermore, there is no indication

that Figure 8b of Wang depicts a triangular or quadrilateral profile. Thus, there is no teaching or suggestion of a triangular or quadrilateral profile in Wang. Accordingly, claims 5 and 6 are in a condition for allowance.

Claims 13 and 14. Claims 13 and 14 are dependent from claim 12, and are in a condition for allowance for at least the reasons set forth above. Claims 13 and 14 also recite in part, “a triangular profile” and “a quadrilateral profile,” respectively. As set forth above with respect to claims 5 and 6, the cited portions of Wang do not teach or suggest triangular and quadrilateral profiles. Accordingly, claims 13 and 14 are in a condition for allowance.

Claims 40 and 41. Claims 40 and 41 are dependent from claim 36, and are in a condition for allowance for at least the reasons set forth above. Claims 40 and 41 also recite in part, “a triangular profile” and “a quadrilateral profile,” respectively. As set forth above with respect to claims 5 and 6, the cited portions of Wang do not teach or suggest triangular and quadrilateral profiles. Accordingly, claims 40 and 41 are in a condition for allowance.

Claims 43 and 44. Claims 43 and 44 are dependent from claim 42, and are in a condition for allowance for at least the reasons set forth above. Claims 43 and 44 also recite in part, “a triangular profile” and “a quadrilateral profile,” respectively. As set forth above with respect to claims 5 and 6, the cited portions of Wang do not teach or suggest triangular and quadrilateral profiles. Accordingly, claims 43 and 44 are in a condition for allowance.

Conclusion

For the foregoing reasons, Applicant submits that all the claims are in condition for allowance.

By responding in the foregoing remarks only to particular positions taken by the Examiner, Applicant does not acquiesce with other positions that have not been explicitly addressed. In addition, Applicant's arguments for the patentability of a claim should not be understood as implying or conceding that no other reasons for the patentability of that claim exist.

Applicant : Somashekar Ramachandran
Subrahmanyam
Serial No. : 10/651,452
Filed : August 29, 2003
Page : 31 of 35

Attorney's Docket No.: 15786-018001

The fees in the amount of \$120.00 for an Extension of Time (One Month) is being paid concurrently herewith on the Electronic Filing System (EFS) by way of Deposit Account authorization.

Please apply any other charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: 11/13/2006

Andrew Leung
Andrew H. Leung
Reg. No. 55,374

Customer No. 26181
Fish & Richardson P.C.
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

Applicant : Somashekar Ramachandran
Subrahmanyam
Serial No. : 10/651,452
Filed : August 29, 2003
Page : 32 of 35

Attorney's Docket No.: 15786-018001

Appendix

Exhibit A

Rossignac, "Issues on Features-Based Editing and Interrogation of Solid Models," *Comput. & Graphics*, Vol. 14, No. 2, pp. 149-172 (1990).

Features and Geometric Reasoning

ISSUES ON FEATURE-BASED EDITING AND INTERROGATION OF SOLID MODELS

JAROSLAW R. ROSSIGNAC

IBM, Research Division, T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598

Abstract—Operations that create additive or subtractive volume features, such as bosses or slots, simplify the computer aided design of mechanical parts. Surface features, whether extracted automatically or selected interactively, group functionally related boundary elements, and thus provide an expedient interface between CAD systems and analysis or manufacturing applications. Despite much progress in CAD, design remains an iterative process and involves error-prone modifications of previous solutions. Features should in principle offer a high level vocabulary for characterizing errors and for specifying how they should be corrected. This paper points out the semantic ambiguities of simplistic feature-based commands for editing models. It recommends procedural models for editing volume features, and corrective volumes for editing surface features. It shows how space decomposition techniques and CSG expressions based on active zones reduce the cost of executing an editing command. Error detection may be automated by supporting intentional features, which correspond to the desired characteristics of the model, and by endowing them with domain dependent validity criteria expressed in terms of associated geometric elements. The paper demonstrates that validity may be tested by simply interrogating a mixed-dimensional geometric structure which is used to represent not only the model, but also the interactions between the geometric elements associated with intentional features.

1. INTRODUCTION

Solid modelling improves the efficiency of the design process for manufactured parts by supporting the geometric representations of these parts. It provides graphic feedback to the designer and offers interfaces to some analysis applications. Despite recent progress in computer hardware and in interactive graphics, geometric design of three-dimensional shapes remains a complex and time consuming task. Since geometric representations of complex mechanical parts tend to be verbose, various abstractions, globally called *geometric features*, are often used to characterize certain types of shapes or to refer to portions of a part model that may be important to the designer or to an application program.

Features are used in Computer Aided Design and Manufacturing for a variety of purposes:

- Geometric features provide a concise description of the parts characteristics [1] and thus simplify group technology and process planning. They also facilitate the communication between designers and solid modelling systems.
- Features provide a mechanism for attaching product or manufacturing information and various attributes to specific parts of a geometric model (see for example [2] for attaching tolerance information to features).
- The properties intuitively associated with common feature types define many convenient shape altering operations [3] that attempt to create features of specified types and dimensions.
- Access to the geometric elements that compose a feature simplifies the interrogation of shape by providing a naming scheme for sets of boundary elements [2], a convenient vocabulary for expressing relevant dimensions and positions [4] and for for-

mulating validity checks that assess the compliance of the model with the designer's intent.

- Expressing and performing engineering changes or simply corrections of design errors may be eased by using geometric features [5]

This paper focuses on the last two issues, namely the use of geometric features to automate—or at least simplify—the editing of the solid model and the checking of the model's compliance with functional requirements (validity conditions).

Most CAD models of 3D manufactured parts are created by combining and incrementally modifying simple models. These combinations and modifications are often conveniently expressed in terms of Boolean operations. The primitive shapes from which the models are constructed are often restricted to arbitrarily positioned and sized solid primitives (blocks, cylinders, spheres ...), generic volume features (holes, slots, bosses ...), and linear or circular extrusions of 2D regions. The resulting part models can thus be represented by a CSG (Constructive Solid Geometry) tree [6], which leads to certain algorithmic advantages (see [7-9] for examples) and to an obvious archival conciseness.

CSG expressions may be complex, and the end-user often prefers to interact with a boundary model, which is algorithmically derived from CSG [10] and contains the list of faces and their adjacency graph [11]. Therefore, it is important to develop techniques for interactively specifying validity conditions and modifications in terms of boundary elements (faces, edges, and their incidence graphs) rather than in terms of CSG. Domain-dependent features provide a particularly convenient vocabulary for accessing relevant groups of boundary elements. On the other hand, direct boundary editing is error-prone, and editing operations, even if specified in terms of boundary information,

should be translated into mathematically well-defined (nonambiguous) operations, such as Boolean set operations or global rounding and filleting operations [12]. Besides, it is important to maintain a CSG representation of the model and to express validity conditions in terms of CSG so that the model can be parameterized, easily edited, and reused.

This paper studies the translation process, which takes validity conditions or model modifications expressed in terms of features (and thus of boundary elements) and performs the appropriate model modifications using CSG operations. Due to a lack of formalism of the semantics of feature-based specification, automatic translation remains a challenging research goal (some pitfalls of a "naive" translation process are pointed out in Section 3). Several new or recently developed techniques are discussed, which do not always provide the correct translation, but at least increase the designer's vocabulary or can automatically generate a tentative solution, which may have to be further adjusted by the designer. Furthermore, the paper addresses the issue of efficiently performing the model modifications or the validity tests by using informationally rich geometric structures and properties of Boolean expressions.

The paper is organized as follows:

- The basic concepts and terminology are introduced in Section 2.
- Section 3 points out some of the limitations of a straightforward use of features for editing and interrogating solid models. This section focuses on concepts and techniques rather than on their historical evolution or implementation. (For a more formal survey of the literature on features, the reader should refer to [13, 14].)
- The importance of procedural models for capturing the designer's intent into a flexible parameterized sequence is emphasized in Section 4. To edit a feature explicitly created by an operation, it may be simpler to "ask" the feature what operation created it, change the parameters of that operation, and reexecute the entire sequence.
- Reexecuting the entire sequence amounts to evaluating the boundary of a CSG representation, and may be very costly for large CSG models. A new approach that reduces the reevaluation cost is presented in Section 5. It derives a CSG expression for the regions that must be added to, or subtracted from, the solid model. Furthermore, Section 5 also presents a recently developed mixed-dimensional geometric representation called SGC (Selective Geometric Complex). Algorithms for SGCs generate a subdivision of space imposed by the features. This subdivision can be used to reduce considerably the amount of geometric calculations and of logic expression evaluations necessary to perform the feature-editing operations.
- Some features do not correspond to a single operation, and it may be too complicated to identify all the operations in the sequence that must be edited

in order to rectify an "invalid" feature. Section 6 demonstrates on some simple examples how corrective volumes, obtained by extruding feature faces, can be used to perform simple feature alterations without reexecuting the design sequence. In more complicated situations, these corrective volumes must be trimmed before they can be added to—or subtracted from—the model of the part. Without the trimming step, side-effects may appear, especially when several features interact or when compound features incrementally created by successive operations, are edited. Trimming is best performed using Boolean operations, but producing a trimming CSG expression may prove difficult and remains the designer's responsibility.

- Section 7 addresses the problem of feature validity. Specifically, it shows how features may be efficiently tested by interrogating the corresponding SGC representation.

2. BASIC CONCEPTS AND TERMINOLOGY

This section clarifies the distinction between intentional features and their geometric embodiment, and between volume features and surface features. It also introduces the CSG notation used in this paper.

2.1. Intentional features and their geometric embodiment

A distinction should be made between *geometric features* and *intentional features*. A geometric feature is a collection of geometric elements (for example, faces or volumes) that form a subset of the part's interior, boundary, and/or complement. An intentional feature [15] is an abstraction for accessing groups of geometric elements and for associating with them a type and consequently certain properties defined for all the features of this particular type. For example, an intentional feature of type *slot* may be associated with a part model. This association indicates that the designer intends to have a slot in the model, *i.e.*, a void bounded on three sides by faces of the model. The intentional feature contains references to the corresponding faces. However, due to model manipulations, the referred faces may have been modified or even deleted from the model's boundary. Whatever remains of them and of the associated void constitutes a geometric feature that may no longer exhibit the properties associated with a *slot*.

Inconsistencies between intentional features and the actual geometry of the part are avoided by treating intentional features only as hints and by relating them to geometric elements through collections of unevaluated references. It is acceptable that some, or all, of these references do not correspond to any geometric element of the model's boundary at some particular stage during the design process. Even if all geometric elements referenced in an intentional feature are present in a geometric model, their shapes and positions with respect to the rest of the model need not comply with the characteristics usually associated with the particular feature type. For example, an intentional

feature of type cylindrical hole could be associated with geometric elements (faces) that have been removed from the model's boundary by some Boolean operation, and thus do not correspond to a "valid" hole. In such situations, the intentional feature is said to be invalid, but should not be discarded, because the designer may have produced (intentionally or not) temporary situations, or instances of the model, where many previously defined intentional features are invalid. The overall validity may later be restored by repositioning a subsolid or adjusting a parameter. The designer should not be required to redefine all intentional features that went through an invalid transition stage.

Furthermore, feature validity is very subjective and in fact depends on the role the feature plays with respect to a particular application. To take a simplified example, a cylindrical hole is a valid "detail feature" to be discarded for analysis purposes only if its radius is sufficiently small; on the other hand, it is a valid "manufacturing feature" for process planning only if it is empty and accessible. Feature validity criteria may be expressed in terms of validity rules, which are logical predicates defined in terms of the referenced geometric elements and of their existence, shape, and relation to other geometric elements of the model. Evaluating the model's geometric references is therefore necessary to establish the validity of an intentional feature with respect to any one of the instances (or stages) through which a solid model evolves during the design process. Consequently, the interrogation of invalid features plays an essential role in correcting design errors [4]. Typically, the presence of an intentional feature of a certain type, valid or not, implies some intention that the designer has regarding the functionality of some portion of the part. Thus, intentional features may provide important hints for model alterations and manufacturing process planning.

2.2. Constructive Solid Geometry (CSG)

CSG (Constructive Solid Geometry) [6] refers to an unevaluated representation scheme for solids obtained by combining, in Boolean expressions, simple primitive shapes of arbitrary dimensions and positions. Solids specified in that way may conveniently be represented by a binary tree whose leaves correspond to primitive shapes, whose internal nodes correspond to Boolean operations and represent subvolumes, and whose root represents the final solid. Often, the primitive shapes

are expressed as the intersection of closed half-spaces. Commonly used half-spaces (planar, cylindrical, spherical) are mathematically defined as the set of points for which the value of a simple linear or quadratic function is negative or null. For practical implementation reasons, solid models are often restricted to be *r*-sets (a subclass of closed, bounded three-dimensional sets with no dangling boundary elements and with a finite number of faces and edges) [16]. They are often represented in terms of their boundary, i.e., a list of their faces (in turn defined in terms of their bounding edges) often structured in an adjacency graph [11]. To guarantee that results of Boolean operations are *r*-sets, a regularized version of these operations is used. It performs the standard operations and then removes the dangling and interior faces and edges and makes sure that a valid boundary is part of the pointset. Theoretically, this cleaning operation amounts to taking the topological interior of the pointsets produced by the Boolean operation (this eliminates the exterior dangling faces, edges, and vertices), and then the topological closure of the result (which amounts to putting a tight boundary around the pointset). These transformations are illustrated in Figure 1. In practice, the faces and edges of the model are classified using neighborhoods [10]. Only the elements that play the appropriate role in the boundary of the solid are kept. Throughout this paper, all Boolean operations are regularized, unless explicitly specified otherwise.

The regularized Boolean operations will be denoted \cup for the union, \cap for the intersection, $-$ for the difference, and \oplus for the symmetric difference. Furthermore, the regularized complement of any solid X will be denoted \bar{X} . For simplicity of notation, it is assumed that the Boolean operators in Boolean expression are ranked by decreasing priority as follows: complement, intersection, difference, symmetric difference, and finally union.

Throughout this paper it is assumed that the part models, or solids, are created by a sequence of operations that add or subtract material or move and combine subsolids through Boolean operations. Therefore, a CSG representation of such a model always exists, even though the explicit construction and use of the CSG tree may be avoided in certain cases.

2.3. Volume and surface features

An important distinction pointed out in [13] separates surface features, which are collections of faces of

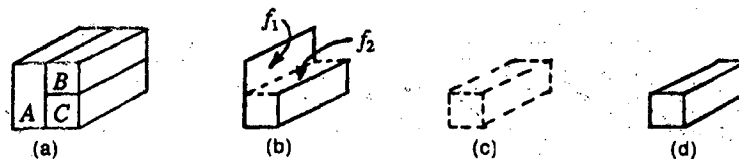


Fig. 1. Regularization: Three blocks, A, B, and C, shown in (a) are combined through a nonregularized Boolean expression $(A \cap B) \cup (C - B)$ to produce the pointset shown in (b), which has a dangling face f_1 and a missing face f_2 . A regularized version of the pointset can be obtained by taking its interior, which is an open set depicted in (c) that does not contain any of its faces, and then taking the closure of the result and thus adding to the model all its faces, shown in (d).

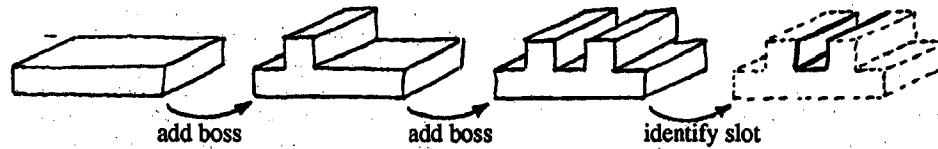


Fig. 2. Surface feature: Adding two bosses (volume features) to the part (left) creates a slot (geometric feature) that can be recognized by the user and interactively associated with an intentional surface feature for later references.

a part model [17–19], such as the walls and the floor of a slot, from volume features, which are full-dimensional pointsets of the part or of its complement, such as bosses and holes [20, 21]. Very rarely are both types simultaneously supported in the same modelling system (as they are in the prototype system MAMOUR [4]). Pratt [13] provides a detailed discussion of the historical motivations, current merits, and drawbacks of both types of features and concludes that all surface features should be converted to volume features, which, although slightly more complex to support, offer greater flexibility for interactive editing and more information for driving analysis and application programs. The author believes that both volume and surface features are useful for editing and that a volume representation of a surface feature need not always be derived.

2.3.1. Volume features. Design is often done in an incremental manner, by first laying out the overall shape, and then adding or modifying details by creating or editing features. The creation of a geometric feature is necessarily accompanied by a modification of the volume occupied by the part and in practice always corresponds to either an addition or a subtraction of material. This transformation may be expressed as the union or difference between the part and the volume feature. Because the volume feature may be viewed as a sophisticated parameterized primitive shape, this approach is particularly effective in dual modellers which derive a boundary representation from a CSG tree.

The computational expense of explicitly deriving the effect of a Boolean operation [10] has discouraged certain developers of solid modellers from evaluating the boundary of the part obtained by subtracting or adding a volume feature. Instead, implicit features (also called unevaluated) have been recommended [19].

A boundary representation of the feature is directly derived from the designer's specification without checking if this representation is geometrically correct.

For example, an implicit feature of type slot may have been defined by mistake as hanging in air, away from the part, or buried inside the part and not accessible from any side. This incompatibility problem does not occur when intentional features are used instead of implicit features because intentional features, although unevaluated, carry no assumption as to their corresponding geometry embodiment.

2.3.2. Surface features. The volume features resulting from shape modifying operations do not always provide a sufficient set of abstraction tools for interacting with the model. For example, a slot feature of interest for manufacturing applications may have been created as a side effect of adding two parallel boss features (Fig. 2). The slot may provide a convenient abstraction for expressing engineering changes (which, for example, modify its width) and thus should be made accessible to the designer through an intentional feature. The use of such a posteriori identified features requires the association of intentional features with a subset of an existing geometry. Often such association is done by interactively selecting a collection of faces of the part model and treating it as a surface feature.

Information provided by surface features may be sufficient for some applications, such as planning for surface finishing operations or as specifying and analyzing dimensions and tolerances [2]. Other applications, such as assembly or manufacturing planning, heavily rely on the manipulation of volume features [22]. Except for simple cases, the derivation of a volume feature that corresponds to a surface feature remains an open issue [13], because there is no unique mapping from surface features to volume features. Typically a selected set of closing faces is added to a surface feature in order to produce a valid two-dimensional shell that unambiguously defines a volume (Fig. 3). Desirable, or even correct, closing faces may not always be obtained by extending existing adjacent faces (Fig. 4). Methods or heuristics for automatically con-

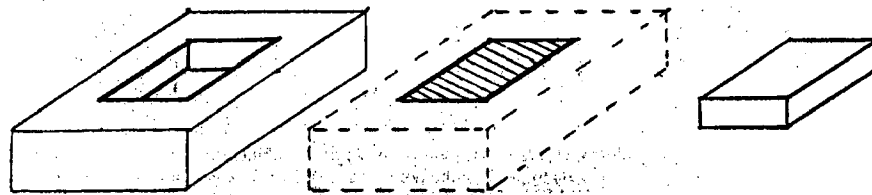


Fig. 3. The volume of a surface feature: By adding a closing face (center) to the faces of a surface feature (left), one obtains a valid boundary of a corresponding volume feature (right).

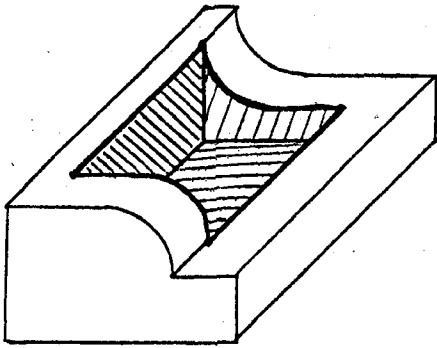


Fig. 4. Complex closing faces: No simple set of closing faces exists for the pocket surface feature.

structing such faces are currently limited to simple situations, and the designer's intervention may sometimes be required to generate acceptable solutions.

2.4. Compound features

Several volume or surface features may overlap. For example, two orthogonal slots (volume features) may have a common intersection volume (Fig. 5).

Similarly, a boss and the adjacent slot may share a common vertical wall (Fig. 6). Furthermore, interior features, such as a boss on the floor of a slot (Fig. 7), may be used as modifiers of other features. It is not always necessary to capture such feature interactions explicitly in the data structure. For example, a geometric element (face or volume) may be shared by several intentional features that are independently used by different applications. On the other hand, a hierarchical organization of intentional features may be useful to represent explicitly compound features (Fig. 8) and also patterns of features (Fig. 9) when such situations reflect the designer's intentions or are important for applications such as process planning. The nature of the geometric and topological interaction between the individual features of a compound feature should be derived, when needed, from the actual geometry of the faces referenced by the individual features.

3. Pitfalls

Because they provide an intuitive, domain dependent, high-level vocabulary, both volume and surface features are good candidates for facilitating the speci-

fication of shape modifying operations and the expression of validity Conditions, provided that one can make the specification convenient and unambiguous.

For the designer's convenience, these specifications have to be unambiguous, so that the effect of shape editing commands can be clearly understood and easily predictable, and the validity rules must precisely characterize invalid situations independently of the verification procedures employed. They also must be convenient, so that the specifications correspond to powerful high-level operations that produce the desired effect and so that validity rules are simple to formulate and powerful enough to trap common design errors. Furthermore, procedures for executing the shape modifying commands and for evaluating validity rules must be available.

This paper shows how extensions of several techniques may be integrated to improve the specification and the execution of unambiguous shape modifications using compound or isolated volume or surface features. It also shows how a rich geometric representation scheme can be used to simplify the expression and evaluation of validity rules. Most of these techniques have been made possible by recent developments in geometric modelling, which must now be integrated. These developments will be briefly summarized, and their potential applications to feature-based editing of solid models will be demonstrated.

3.1. Limitations of simple shape modifying techniques

To stress the need for the approaches such as those proposed in this paper, this section discusses the limitations of several simple schemes that come to mind as possible ways of using features to modify and test solid models.

3.1.1. Implicit features. Formerly mentioned implicit features may be trivially edited by modifying their parameters. For example, an implicit slot can be moved and enlarged by changing its position and its width. However, as pointed out earlier, the existence of an implicit feature with specified dimensions and position does not guarantee that the corresponding geometric feature with the expected characteristics is to be found on the part. Thus, to produce a reliable description of a part, implicit features should be treated as intentional features, and the corresponding geometric features should be constructed (if possible) and their validity (i.e., compliance with functional requirements) assessed.

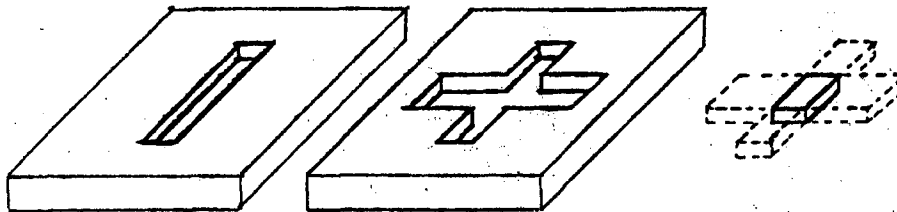


Fig. 5. Two interfering features: Subtracting a slot (volume feature) from the model (left) that already has a slot feature creates a model (center) in which the volumes of the two features interfere (right).

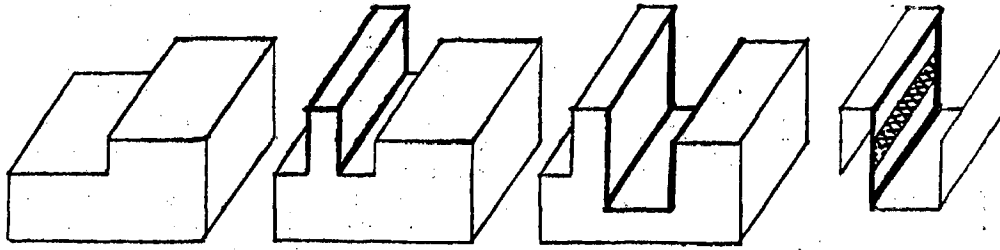


Fig. 6. Shared face: Adding a boss (volume feature) to the model (left) and then subtracting a slot feature (center) creates a model in which the boundaries of two adjacent features overlap along a portion of a face (right).

3.1.2. Procedural models. To further automate the generation of models and allow the designer to concentrate on high-level design decisions, it is suitable to support the automatic derivation of CAD models from a set of functional constraints specified by the designer. The functional requirements specifying the geometric characteristics of intentional features could be considered as constraints and combined with the geometric constraints describing the part. A constraint solving system would converge to a valid solution, if such a solution exists, or declare that the specification (i.e., set of constraints) is invalid. Such a scheme would have the considerable advantage of supporting incomplete specifications of features. For example, an intentional feature of type slot could be defined and its dimensions specified, but its position would not be provided by the designer, except for one constraint: The slot should be abutting on a given face of the object.

Such an automated approach has been given a serious consideration [23,24], but it carries a high computational cost and requires that designers provide a complete set of consistent constraints before the CAD system can create a model and return some useful feedback. Deriving and maintaining such systems of constraints, especially when additive (bosses) and subtractive (pockets, slots, holes) features interfere is a considerable endeavor, as pointed out in [25]. Furthermore, a simple indication that the set of constraints is incompatible does not provide useful hints as to what part of the specification should be modified to produce the correct result. A more practical approach is to build models incrementally by transforming and combining simpler models. A procedural rather than declarative approach, in which the designers specify a sequence of

operations that transform a model in attempt to satisfy constraints, has been described by Rossignac in [26]. It relies on the designer's ability to decompose the problem into an ordered set of subproblems that can be solved one at a time. The procedural specification (i.e., the sequence of operations that solve the individual problems) is saved and can be edited by the designer and reexecuted on demand. This technique could be used for feature-based editing by considering the implicit features as intentional features to be created in a predefined order. Reexecuting the specification would attempt to create the geometric counterpart of the intentional features at specified positions and would report whether the creation was successful (i.e., whether valid features have been produced).

Considering that a procedural model can be obtained simply by storing the designer's commands into a log file and making the file available for modification and reexecution fails to address three important problems:

1. Features successfully created during an execution of the procedural model can be invalidated by the subsequent creation of other features. Therefore, to assess the validity of a design, intentional features created at an early stage of the specification must be preserved and methods for accessing the corresponding geometric elements (whenever they exist) and for testing the compliance of these elements with feature validity rules should be available.
2. Feature parameters may be defined in terms of other features. For example, a boss may be intended to lie at the center of the floor of a rectangular pocket. Although this relation may have been established at the creation of the boss, subsequent editing of

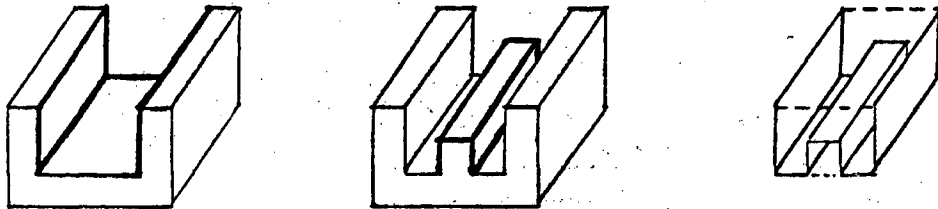


Fig. 7. Nested features: Adding a boss feature in the middle of a slot feature (left) creates a model (center) with a nested feature (right): The boss is inside the volume of the slot.

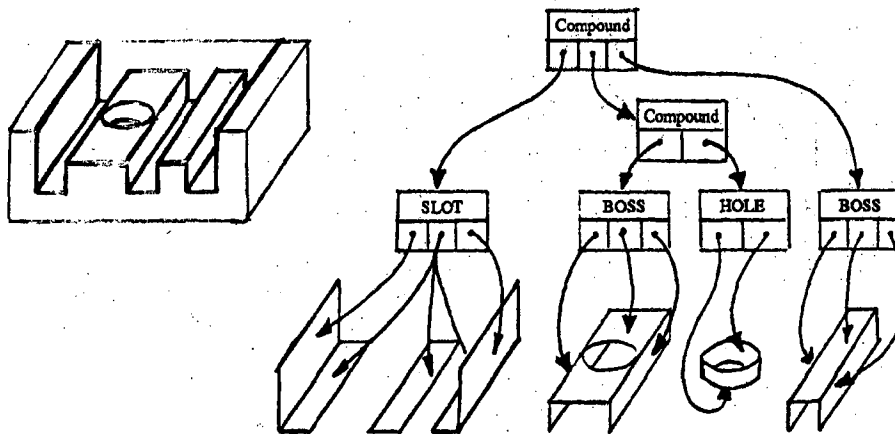


Fig. 8. Compound features: A hierarchical organization of features may be used to represent a compound feature made of a slot that contains two bosses, one of which has a hole (left). These relations can be captured by creating additional intentional features of type compound that refer to other features instead of referring to geometric elements (right).

the procedural model may alter the position of the pocket. To preserve the relation between the positions of the two features, this relation must be captured in the procedural model and used to position the boss at each execution.

3. Surface features are typically selected by the user in an interactive mode, preferably using a graphic cursor to pick the appropriate geometric elements from a picture of the model on a computer screen. Surface features in a model may be used to attach tolerances or other manufacturing attributes to particular portions of the model and to serve as clues for process planning [5], or simply to provide constraints on the position and size of volume features to be created later by the procedural model. It is therefore essential that surface features, once selected by the designer on one instance of a model, be selected automatically when a new instance of the model is created. To support automatic reselection, operations that select surface features must be captured in the procedural model in such a manner that their execution produces the desired result, even when earlier parts of the procedural model are modified.

Techniques for supporting procedural models together with intentional surface features and for capturing relations between such features have been developed and implemented by Borrel, Nackman, and the author, and are described in [4]. They will be briefly reviewed in Section 4 and their applications to feature-based editing will be discussed.

3.1.3. *Local boundary modifications.* If no procedural model is available for editing and reexecuting, or if the execution of the procedural model is costly, the part model may sometimes have to be directly edited, or "patched." Since a valid solid model is unambiguously described by its boundary, boundary "tweaking" seems an attractive technique for editing. For example, the four vertices of the floor of the slot in Fig. 10 could be raised to change the depth of the slot. A new boundary representation would be readily available if the floor and the adjacent faces were implicitly defined by their vertices. However, such boundary tweaking techniques may require major alterations of the boundary structure of the object. If polyhedral modellers, if the geometry of the edges and faces of the model are implicitly represented in terms of vertex coordinates, a face (for example, the floor of

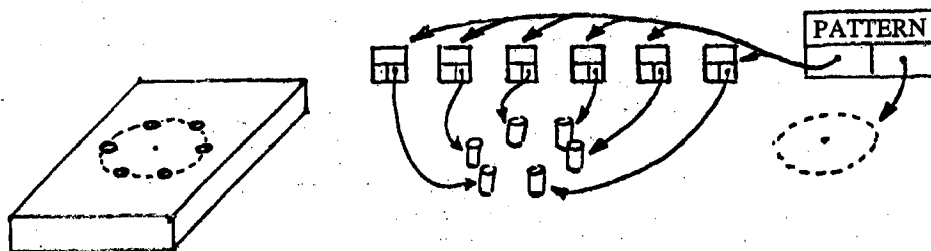


Fig. 9. Patterns of features: A pattern of hole features (left) may be represented by a compound feature referencing the intentional features of each hole (center). The compound feature has a description of the pattern parameters (right) and can be used to access and interrogate the entire pattern.

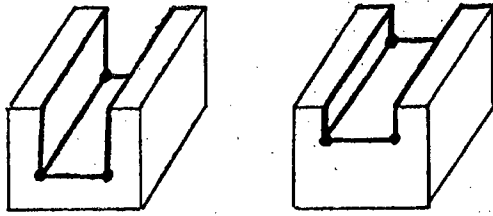


Fig. 10. Boundary tweaking: The floor of a slot (left) may be raised by moving up its vertices (right).

a pocket) may be moved simply by displacing its vertices. However, the vertices of each face must remain coplanar. Coplanarity is guaranteed by forcing the vertices of the face being moved to remain coplanar and to move along extensions of the edges they bound.

The situation is more complicated when vertices correspond to intersections of curved surfaces. Suppose that the designer wishes to move only one face of the feature and wants to express this modification in terms of vertex displacement. Then vertices are constrained to move along extensions of abutting (curved) edges while simultaneously remaining on the edited surface as it evolves. The number of degrees of freedom, which specifies how many of the vertices may be moved independently, is dictated by the nature of that surface and its acceptable deformations (scaling, rotations ...).

In addition, the validity domain, which specifies the maximum amount of vertex displacement along each edge so that the boundary structure (adjacency graph between faces) remains constant, is defined by the geometries and relative positions of the faces. Restricting the displacement of each vertex along the extension of an edge so that it does not exceed the intersection of this extension with other faces is not sufficient to guarantee that the resulting boundary will not be self-intersecting (as is the case in Fig. 11).

An alternate approach is to modify the boundary structure or to alter several faces simultaneously. A simple example may be found in Fig. 11, where a vertical left wall is added to connect the lowered floor to the abutting cylindrical face, but in general it is difficult to devise procedures for specifying these modifications. Furthermore, detecting self-intersecting boundaries is

computationally difficult, and no unambiguous and useful semantics has been defined for specifying how self-intersecting boundary representations should be corrected.

In conclusion, the semantics of boundary tweaking operations is not well defined and the results may be invalid. This paper thus proposes techniques that use mathematically well-defined set theoretic operations to alter features.

A possible approach would be to convert incorrect surface features into volume features (by the insertion of closing faces) and then delete these volume features and create new correct ones. As pointed out earlier, the set of closing faces is not unique for any surface feature, and even a single suitable set may be hard to produce. To overcome the "closing face problem," a technique based on corrective volumes, previously employed by Requicha and the author for local blending operations [12], will be proposed in Section 6. Here, it uses a generalization of sweeps or extrusions to create solids that can be added to—or subtracted from—the part in order to change the dimensions or positions of geometric features.

3.1.4. Order dependency of volumetric alterations

A volume feature, whether directly created by a Boolean operation or derived by closing a surface feature, could in principle be modified by deleting it and, if necessary, by creating a new volume feature with the desired characteristics (Fig. 12).

Deletion of a volume feature may be obtained by using the inverse of the Boolean operation that could have been used to create it. For example, a subtractive feature of type slot could have been created by subtracting the corresponding slot volume from some previous representation of the part. Therefore, adding the volume back, using a Boolean union, would delete the feature. Unfortunately, this approach suffers from three major problems, which could be qualified as "undesirable side effects."

1. As demonstrated in Fig. 13, the lack of associativity properties of set theoretic Boolean operations do not in general permit to undo the effect of subtracting (respectively adding) a feature by adding (respectively subtracting) it back. Specifically: $(A - B) \cup B \neq A$, unless $B \subset A$, and similarly $(A \cup B) - B \neq A$, unless $B \subset A$.

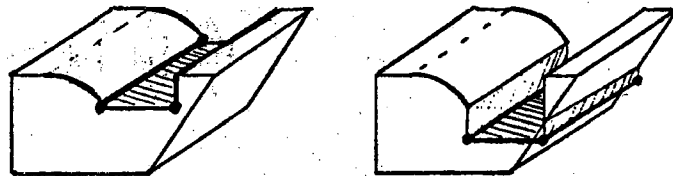


Fig. 11. Validity problems: The depth of the slot (left) is altered by lowering its floor face. In the resulting model, the boundary is disconnected and self-intersecting. To connect the boundary on the right side of the slot, the right wall can naturally be extended to follow the vertices. On the left side, however, the wall is cylindrical and the lowered vertices do not lie on its extension. They either have to be moved horizontally, or a new vertical face must be created. In any case, the boundary is self-intersecting on the right side of the slot, and therefore portions of it must be eliminated, which involves geometric calculations.

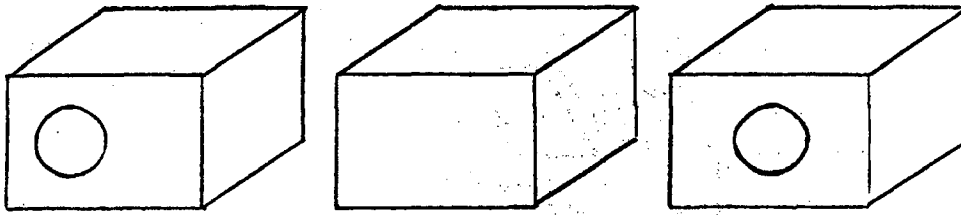


Fig. 12. Editing by deletion and creation: A hole volume feature (left) may be moved by deleting it (center) and by creating a new hole in the right position (right).

2. When additive and subtractive features interfere, the order in which they have been created is important and in general cannot be altered without producing undesirable side effects, such as the one illustrated in Fig. 14.
3. Modifying features by deleting them and creating new ones could modify or destroy other features (see Fig. 15).

For solids created by combining simpler solids and volume features through Boolean operations, correct results may be obtained, without reevaluating the whole Boolean expression, by confining the deletion of the old feature and the creation of the new correct one to the active zone of the old feature. Active zones have been introduced by Voelcker and the author in [27], and would correspond here to the portion of space where the shape of the feature is important. A formal definition and relevant properties of active zones will be summarized in Section 5, and new properties for applications to feature-based editing will be derived.

Instead of computing the active portions of all features (i.e., their intersection with their active zones), space decomposition techniques [28–30] may be used to precompute and store the geometry of the active portions of all features simultaneously, and thus to improve the performance of algorithms that execute the editing operations. Then, each editing operation may be confined to the appropriate regions without repeating expensive geometric calculations each time. Such an approach has been proposed by Pratt [13] using an extension of the radial edge structure developed by Weiler [28]. Section 5 briefly presents a different, more general, and slightly simpler structure called Selective Geometric Complex (abbreviated SGC) described by O'Connor and the author in [29]. Algorithms for sub-

dividing SGCs may be used to decompose space into cells of dimension three or less, such that given any cell C and any volume (or even surface) feature F , C either lies entirely in the interior of F , entirely in its complement, or entirely in the boundary of F (when C is a face and F is a volume feature). Each cell of an SGC “remembers” what features it belongs to and is associated with an attribute which defines whether the cell is part of the represented solid or not. Fig. 16 illustrates such a decomposition. The applications of the SGC structure to the deletion and modification of features within their active zones will be demonstrated.

3.2. Limitations of simple interrogation techniques

Detecting whether the geometry associated with an intentional feature satisfies the validity requirements explicitly associated with that particular feature-type is extremely convenient for validating a design and automatically verifying that changes produced by adding new features or modifying old ones did not create undesirable side effects. However, the validity of each individual feature may not be sufficient to assess the validity of a complex part, and sometimes, a relation between several features is also important. Most of these validity requirements may be explicitly attached as rules to single or compound features [4], and automatic procedures for checking feature validity may be invoked, as described in Section 7.

Some validity rules may be characterized in terms of topological relations between volume features and can be expressed in terms of Boolean operations between a volume feature and the part it is supposed to modify. For instance, when a slot B is to be subtracted from a part model A (Fig. 17 center), one can detect situations where the slot is too deep and where its floor

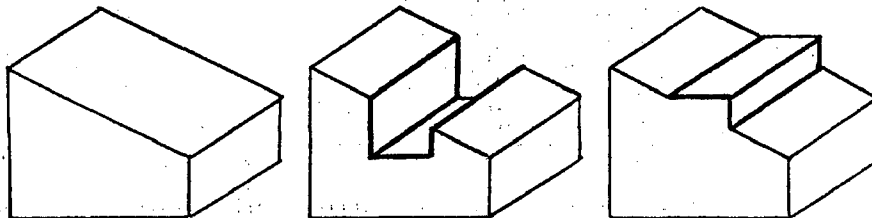


Fig. 13. Side effects: In the slanted top face of the model (left), a slot is created (center) by subtracting a block. Adding the block back (right) does not restore the original model.

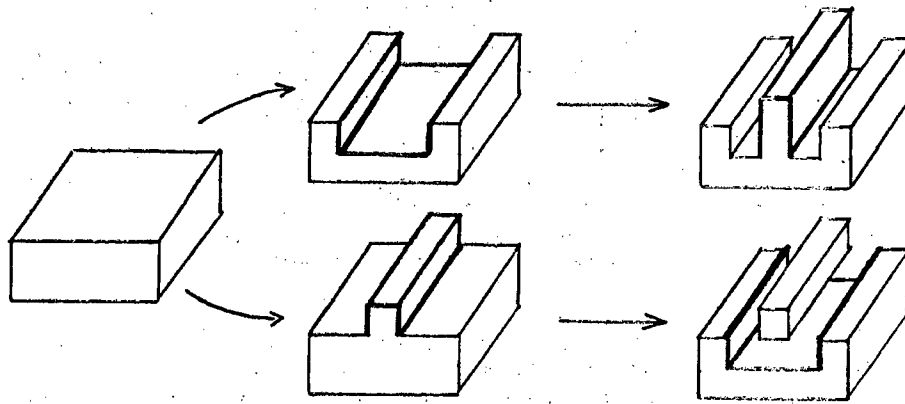


Fig. 14. Order dependency: Modifying a solid (left) by first creating a slot (top center) and then a boss (top right) produces a different result than first making the boss (bottom center) and then the slot (bottom right).

is missing in the boundary $A - B$ (Fig. 17 right) by checking whether $B - A$ represents the empty set or not (this works only if the floor is not coincident with the bottom face of A).

When regularized Boolean operations are used, such simple tests are insufficient to distinguish between significantly different situations. For example, a distinction between the correct situation and the unacceptable configuration in Fig. 18 may not be obtained as previously suggested by considering $B - A$, because $B - A$ is empty in both cases. Clearly, in most cases, a more complex test involving Boolean combinations of auxiliary solids may be provided, but each situation and each test may require different types of auxiliary solids. For example, a "roof," thin block B' over B may be used: $B' \cap A$ is empty[†] in the correct situation of Fig. 19, but is not empty in the invalid case of the same figure. These tests are expensive to perform and may require constructing and intersecting complicated auxiliary volumes. Furthermore, this approach is limited to volume features. An alternate approach is proposed in Section 7 which demonstrates that most com-

mon validity criteria may be tested by querying the existence of lower-dimensional parts in an SGC representation of the space decomposition defined in terms of features.

4. PROCEDURAL MODELS

As pointed out in the introduction, procedural models are particularly convenient for trial-and-error geometric design, because they capture the designer's specifications in a form that can be easily understood, edited, and repeatedly executed. This section briefly presents a prototype system for procedural modelling, called MAMOUR, which has been implemented in the object-oriented language AML/X [32] and described in more details in [4, 5]. MAMOUR supports intentional features with validity rules and can keep track of relations between different features.

4.1. Sequences

Such a system offers an adequate platform for integrating the techniques presented in this paper. As the user of MAMOUR creates and transforms a part model, the system automatically constructs a procedural model composed of an ordered set of operations that measure geometric (or other) properties, create or move primitive or subsolids, define intentional features, or perform Boolean operations between objects. The

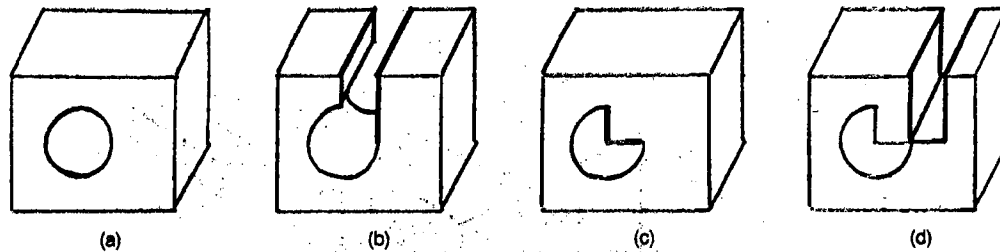


Fig. 15. Unintentional feature destruction: In a model with a through-hole (a), a vertical slot (b) is created in a wrong position. Deleting the slot by a union creates a solid (c) in which the through-hole feature is partly destroyed. Making a new slot in the right position (d) produces a valid slot, but leaves the through-hole invalid.

[†] Testing whether a Boolean combination of two or more solids is empty requires evaluating its boundary or performing a Null Object Detection test on a CSG representation [31, 27].

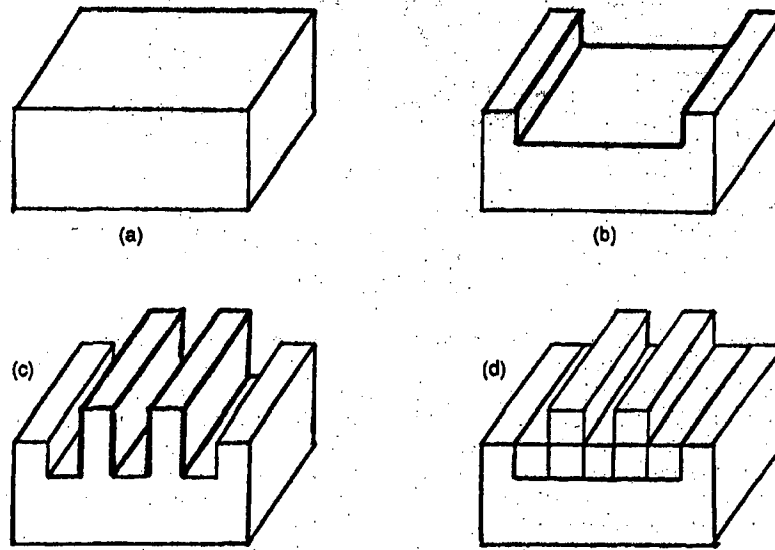


Fig. 16. Space decomposition: In the model (a) a slot is made (b). Then two bosses are created inside the slot (c). Space is decomposed into 3D cells (d), so that each cell is either inside or outside the original model and any one of the features.

execution of the procedural model constructs a geometric model and the associated set of intentional features. The procedural model can be edited by the user without interrupting the design session, because it is stored in memory as an aggregate (list) of AML/X objects.† Each object corresponds to an operation, or design step. Fig. 20 top illustrates one such sequence. Each operation has its own internal variables and execution methods. A whole sequence of operations may be captured in another AML/X object of type SEQUENCE and can be edited, executed, and included as a parameterized macro operation into other sequences.

4.2. Unevaluated parameter expressions

To provide a greater flexibility and multiple applications of the same sequence, each operation and

therefore each sequence is parameterized. When several sequences are pieced together in different ways or when an early part of a sequence is edited, an individual operation may be executed on a model different from the model that has been used to specify that operation. For example, a make-rib operation may have been used to make a rib in the center of the boss in the original model. The parameters locating the rib were derived from the position of the boss. If operations that construct or modify the boss have been edited so that the shape of the boss is changed, the execution of the make-rib operation will produce a rib that is no longer in the center of a boss, unless make-rib can adapt its execution to the new geometry. This adaptability is possible in MAMOUR, because parameters of operations may be stored in an unevaluated form, thus capturing the designer's intents (for example, to position the rib in the center of the boss). To simplify the formulation of this constraint, an intentional feature is associated with the boss. Intentional features in MAMOUR are also AML/X objects with internal variables and methods. The internal variables typically contain sets of unevaluated references to boundary elements of the model (for ex-

† An object is an entity that has a type (for example, "Drill operation"), a set of internal variables (for example, expressions that define the radius and position of the hole), and a set of procedures, called methods (for example, the one that modifies a CSG representation of a part by subtracting a cylinder of the appropriate radius).

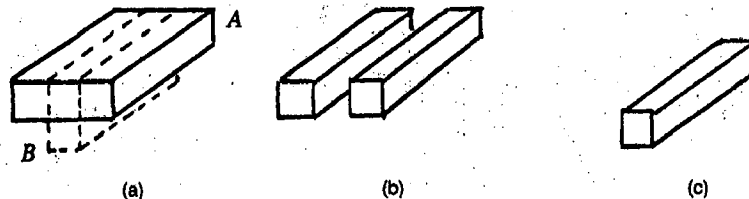


Fig. 17. Validity test: The position of a volume feature B in relation to the part A is shown in (a). Subtracting B from A fails to produce the desired slot feature (b) because the depth of B is too large. Sometimes, such situations may be detected by testing if the difference $B - A$ is an empty solid or not (c).

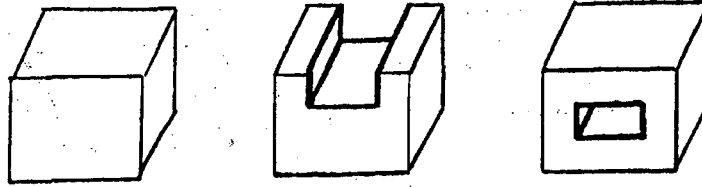


Fig. 18. Inadequate validity test: Subtracting from a part A (left) a correctly positioned slot volume feature B produces a valid geometric feature of type slot (center). Subtracting an ill-positioned slot B creates an invalid geometric feature (right). The difference may not be detected by considering a regularized Boolean combination of A and B.

ample, references to the faces of the boss). The methods can be used to evaluate these references and obtain a geometric description of the appropriate elements (for instance, of the floor face of the boss). These geometric elements are also AML/X objects with methods for computing their geometric properties (for example, the center of a face). Thus, if *boss1* is the name of the intentional feature that corresponds to the geometric feature of type boss, the position of the rib may have been specified in *make-rib* using *boss1.floor().center()*, which defines the correct position, as long as the geometry referenced in *boss1* is a valid boss, or at least has a floor face.

Until now, MAMOUR has been interfaced with only a 2D geometric modeller, and therefore intentional features contain only edge-references. Such an edge-reference is an unevaluated AML/X expression defined in terms of:

- the structure of a CSG representation of the particular model (if available),
- the adjacency information typically available in a boundary representation, and
- the names of operations that created or modified the edge.

Tools for automating the construction of expression that consistently identify faces in various versions of a model are currently under investigation by Paul Borrel and by the author.

4.3. Validity tests

The validity of intentional features, and thus of the produced model, may be tested automatically using validity rules attached to features. To a particular feature may be associated several rules (predicates), which measure geometric and topological properties of boundary elements. If, for example, any rule evaluates to FALSE, the feature is considered invalid. Note that

the same feature may be valid for another application with different validity criteria. A rule can be any expression[†] in AML/X that returns a Boolean value. Typically such expressions involve references to intentional features, and therefore indirectly to corresponding geometric entities, and also calls to methods applied to these entities for computing their properties or deriving specific measures.

4.4. Editing features

4.4.1. *Volume features.* Intentional features can be created in MAMOUR by executing a shape-modifying operation, which attempts to produce a geometric feature through a Boolean operation and at the same time creates an object that represents the intentional feature and contains, in its internal variable, references to the corresponding geometric entities. Note that these entities need not always exist in the boundary representation of the model as it evolves. Since a volume representation of the geometric feature is used to the Boolean operation, these features correspond to the volume features discussed above. Such volume features may easily be used for editing the model. To each boundary element and to each intentional feature in MAMOUR is associated a history attribute, which indicates what operation created the entity. After a face is graphically selected, its history may be accessed by the designer and the parameters* of the corresponding

[†] In particular, it can combine through an OR operation several Boolean subexpressions, and thus provides a simple mechanism for expressing validity rules as conjunctive forms or even more complex Boolean combinations.

* It is the designer's responsibility to decide which parameters should be edited and how. The systems could provide some help by maintaining a dependency graph that relates faces to parameter expression. Often these relations are simple, and we have not investigated such facilities for further assisting the designer.

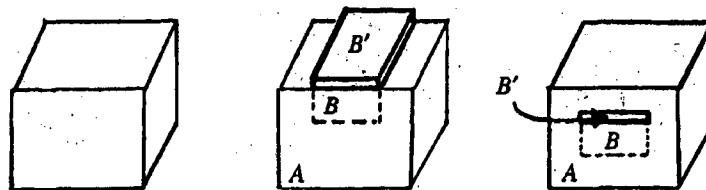


Fig. 19. Auxiliary volumes: A thin block B' over the slot B may be used to test whether the slot B is made in the block A (left) is accessible from the top. In the valid configuration (center) the set $B' \cap A$ is empty. In the incorrect configuration (right) $B' \cap A$ is not empty.

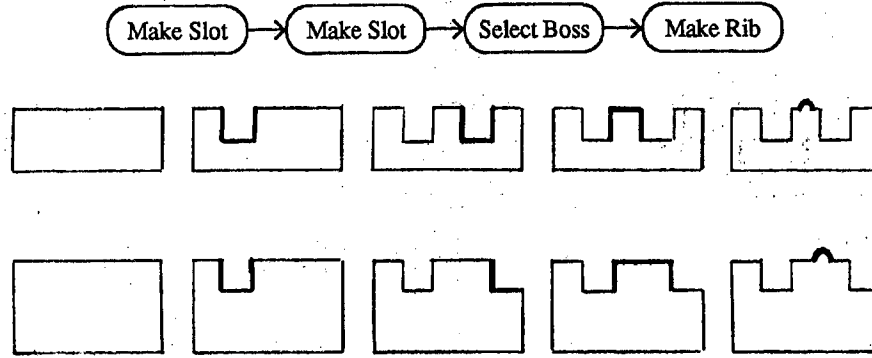


Fig. 20. Sequence of operations: The sequence (top) contains two slot-making operations, which subtract two volume features, followed by a feature-selection operation which identifies a surface feature of type boss used in the last operation to position a rib volume feature. The execution of the sequence produces a model (center). Editing one of the slot-making operations and reexecuting the sequence applying it to a different starting object produces a different result (bottom), which still reflects the designer's intent to center the rib on the boss.

operation edited so that the correct feature is created by reexecuting the whole sequence. This reevaluation may be computationally expensive. Therefore, alternate techniques for editing the model are discussed below.

4.4.2. *Surface features.* Intentional features may also be created without geometric modifications by a select operation which takes as parameters a feature type and a list of references to existing boundary elements. Typically there is no single shape modifying operation responsible for creating such a surface feature and thus no explicit volume representation. Techniques for editing such surface features by adding another operation to the sequence are addressed in the next section. They complement facilities for editing operations already in the sequence, which are well suited for modifying volume features when the cost of reevaluating the entire sequence is not prohibitive.

5. EFFICIENT EDITING OF VOLUME FEATURES

Active zones were introduced by Voelcker and the author in [27] to speed up certain geometric computations over Constructive Solid Geometry (CSG) representations. The active zone associated with a CSG node is the region in which the shape of the set represented by the node is important. An active zone is defined algebraically as the intersection of certain nodes of the whole CSG tree, and thus its CSG expression is always available. The concept of active zones is applied in this paper to deal with editing feature. In this section,

from the properties of active zones, a CSG expression is derived, which, in conjunction with spatial localization techniques, may be used to improve the performance of algorithms that update the boundary representation of a model when a volume feature is altered. These improvements are particularly interesting when a spatial decomposition scheme is used for the boundary representation.

5.1. Active zones

Let S be a CSG representation of a solid encoded in a binary tree. (For simplicity, we shall use the same symbol for the solid and its CSG tree.) Let A be any primitive (or any internal node) of S . The active zone of A in S , denoted Z_A^S , is equal to the Boolean difference $I_A^S - U_A^S$, where I_A^S and U_A^S are respectively the I-zone and the U-zone of A in S . Let A be a primitive or internal node of a subtree N in S , and let F be the parent node of N of another node B . The following properties provide a recursive formulation for incrementally constructing I_A^S , U_A^S , and Z_A^S .

- I_A^S is the universe (Euclidean three-dimensional space) and U_A^S is the empty set.
- When $F = N \cup B$ or $F = B \cup N$, then $I_F^S = I_A^S$, $U_F^S = U_A^S \cup B$, and $Z_F^S = Z_A^S - B$.
- When $F = N \cap B$ or $F = B \cap N$, then $I_F^S = I_A^S \cap B$, $U_F^S = U_A^S$, and $Z_F^S = Z_A^S \cap B$.
- When $F = B - N$, then $I_F^S = B - U_A^S$, $U_F^S = I_A^S$, and $Z_F^S = Z_A^S \cap B$.
- When $F = N - B$, then $I_F^S = I_A^S \cap \bar{B}$, $U_F^S = U_A^S$, and $Z_F^S = Z_A^S - B$.

It follows that CSG expressions for I_A^S and U_A^S , and thus for Z_A^S , may be computed by traversing the path[†] in the tree from A to S , and constructing the Boolean expressions for I-zones and U-zones using intersections

[†] Automatic extraction of existing 3D features is a difficult and expensive process [33], especially when loose conditions are used to identify features. For example, a slot with a filleted bottom edge may still be a slot, although the floor face is not directly connected to the wall. In MAMOUR, references to edges of features may be specified in a semiautomatic way using graphic interaction. These references are then integrated into unevaluated expressions, stored with the model, that will identify the corresponding feature in a family of part models generated by reexecuting a modified version of the sequence.

[†] The path is defined as the set of nodes traversed by moving from S to A in the CSG tree, following the parent-to-child links.

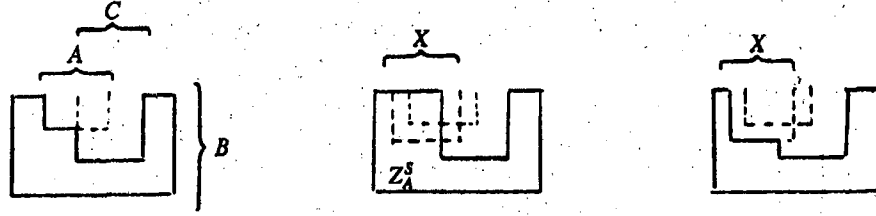


Fig. 21. Changes are confined to the active zone: Replacing the primitive A by X in the CSG definition of S (left) can only affect S in the active zone of A (center). The result is shown (right).

with other branching nodes" or with their complements. On the other hand, these expressions need not be explicitly constructed, nor stored, for most applications, since efficient algorithms presented in [27] can perform calculations with respect to I_A^S and U_A^S using the appropriate branching nodes in the original tree S.

Several applications of Active Zones are discussed in details in [27]. For instance, it is shown that if a node A does not intersect its active zone, it can be replaced by the empty set. Similarly, if a node A contains its active zone, it can be replaced by the universe without affecting the set represented by the whole tree. (These results provide a new algorithm for testing whether any particular node is redundant.) Algorithms for the detection of interferences among solids defined in CSG and for the generation of shaded pictures directly from CSG by ray-casting[34] or depth-buffering[9] are based on facilities for classifying subsets of the boundary of primitives against the CSG tree. The performance of these algorithms may be improved using only the I-zones of the primitives, and not the whole CSG tree for the classification.

5.2. Editing volume features

5.2.1. Localization to the active zone. Because a volume feature corresponds to an internal node of the CSG tree, it has an active zone, an I-zone, and a U-zone. It was established in [27] that all changes to a primitive or node A inside its active zone Z_A^S will affect the shape of S and conversely that changes to A outside of Z_A^S will leave S unchanged. This property and related properties are used in this section to produce trimming expressions for localizing changes that implement a volume feature modification. For example, let A be a volume feature subtracted in the sequence $(B - A) - C$ defining a solid S (see Fig. 21); its active zone Z_A^S is $B - C$. Only changes to A inside $B - C$ need to be taken into account.

5.2.2. Localization to the altered portion of the feature. When a volume feature A is edited and replaced by X in a CSG representation of the solid S, a new boundary may be obtained without reevaluating the

entire boundary of the solid because changes are restricted to $A \oplus X$, the symmetric difference[†] between A and X [35]. Consequently, in the example Fig. 21, only alterations in $(A - X) \cup (B - C)$ are needed in order to compute the boundary of the solid obtained by replacing A by X in the CSG tree of S.

5.2.3. Combined localization. Let o and w represent respectively the empty set and the universe. Let S_X denote the set represented by the tree of S in which A was replaced by X. The two previous results may be combined to yield: $S_X = S \oplus Z_A^S \cap (X \oplus A)$. A proof of this equation may be derived from Equation (4) in [36], given that $(X \oplus Y = Z) \iff (X = Y \oplus Z)$ holds for any three sets, X, Y, and Z.

5.2.4. Classifying additive and subtractive parts. The symmetric difference $A \oplus X$ may be decomposed into two disjoint sets, $A - X$ and $X - A$, which can be treated separately. For example, when a negative (subtractive) volume feature A is replaced by X, a subset of $A - X$ must be added to S and a subset of $X - A$ must be subtracted (see Fig. 22 for an example). This section derives new CSG expressions that characterize these subsets and are simpler than the CSG expression of s§.

Although Z_A^S is the smallest region where changes to A affect S, the classification of $A - X$ and $X - A$ against the CSG expression of Z_A^S in general contains unnecessary steps which can be eliminated using the following property:

$$S_X = S - (A_{\bar{X}} - U_A^S) \cup (A_X^+ \cap I_A^S),$$

where A_X^+ is the portion of material added to A and where $A_{\bar{X}}$ is the portion of material subtracted from A during the same operation.

First consider the case of an additive feature A. We have: $A_X^+ = X - A$ and $A_{\bar{X}} = A - X$. For a proof, first we establish the following:

[†] The symmetric difference between A and X is defined as: $A \oplus X = (A - X) \cup (X - A)$.

[‡] The term disjoint is used here for three-dimensional volumes whose intersection is empty or of lower dimension.

[§] Note that each one of these CSG expressions may be further trimmed down by removing all primitives that are disjoint from $A - X$ (respectively $X - A$) by using techniques discussed in [37].

[‡] The symbol B in the above properties refers to what is formally called a branching node of A in S, i.e., a node that does not lie in the path from A to S, but whose parent node does. I_A^S and Z_A^S are defined as intersections of such branching nodes or of their complements.

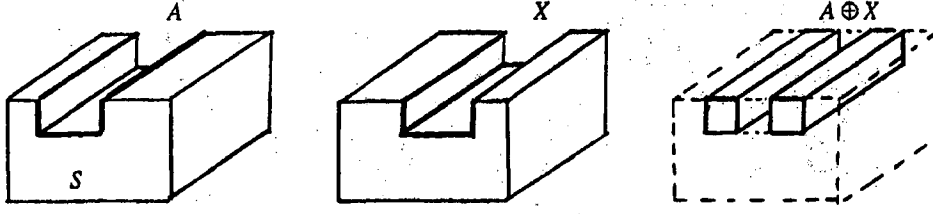


Fig. 22. Changes are confined to the symmetric difference: The solid S (left) is represented by a CSG tree defined in terms of a subtractive volume feature A. Replacing A by X in the CSG tree of S may change the shape of S (center) only in the symmetric difference $A \oplus X$ (right). Note that $A - X$ is added and that $X - A$ is subtracted.

$$I_A^S \cap U_A^S \subset I_A^S \cap U_A^S \cup S_o \cap U_A^S,$$

$$I_A^S \cap U_A^S \subset (I_A^S \cup S_o) \cap U_A^S,$$

$$I_A^S \cap U_A^S \subset S_w \cap U_A^S, \text{ by Proposition 5 of [27],}$$

$$I_A^S \cap U_A^S \subset S_o, \text{ by Proposition 6 of [27].}$$

Now the proof may be constructed as follows:

$$S_X = S_o \cup (X \cap I_A^S), \text{ by Proposition 1 of [27],}$$

$$S_X = S_o \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S),$$

since $I_A^S \cap U_A^S \subset S_o$ (see above),

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup S_o \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S),$$

since $S_o \cap \bar{A} \cup S_o \cap X \subset S_o$,

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S_o \cap U_A^S) \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S),$$

since $S_o \subset U_A^S$ (see [27]),

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S_o \cap U_A^S) \cup (A \cap X \cap I_A^S \cap \bar{A} \cap X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S),$$

since $A \cup \bar{A} = \psi$,

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S_o \cap U_A^S) \cup (A \cap I_A^S \cap \bar{A}) \cup (A \cap X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S) \cup (\bar{A} \cap X \cap I_A^S),$$

$$S_X = (S_o \cup A \cap I_A^S) \cap (\bar{A} \cup X \cup U_A^S) \cup (\bar{A} \cap X \cap I_A^S),$$

factoring out S_o and $\bar{A} \cup X$,

$$S_X = S \cap (\bar{A} \cup X \cup U_A^S) \cup (\bar{A} \cap X \cap I_A^S),$$

replacing X with A in Proposition 1 of [27],

$$S_X = S - ((A - X) - U_A^S) \cup ((X - A) \cap I_A^S),$$

using complementation.

Now note that for a subtractive feature, one may consider the complements of A and X to be additive features, and the same proof holds.

Since the interiors of A_X^+ and of A_X^- are disjoint, the interiors of $A_X^+ - U_A^S$ and $A_X^- \cap I_A^S$ are also disjoint. Therefore, since \cup and $-$ are regularized, we also have:

$$S_X = S \cup (A_X^- \cap I_A^S) - (A_X^+ - U_A^S).$$

Consequently, subsets of A_X^+ need only be classified against I_A^S ; subsets inside I_A^S should be added to S, while subsets outside may be discarded. Similarly, subsets of A_X^- need only be classified against U_A^S ; subsets outside U_A^S should be subtracted from S, while subsets inside U_A^S may be discarded.

The CSG expression of Z_A^S , defined as $I_A^S - U_A^S$, is a complex as the combination of the CSG expressions for I_A^S and for U_A^S , and classifying a point with respect to Z_A^S may often require classifying it against I_A^S and against U_A^S and then combining the result. The above result suggests that A_X^+ needs only be classified against I_A^S and not against both I_A^S and U_A^S . Similarly, A_X^- needs only be classified against U_A^S and not against both I_A^S and U_A^S . Significant improvements for algorithms that perform feature modification result. To further improve the performance of these algorithms, tree pruning and other space subdivision techniques [39, 40, 35, 41] may be used in conjunction with classification against I_A^S and U_A^S . Note, however, that this speed-up may suggest to add to Sportions of A_X^+ that are already included in S, or to subtract from Sportions of A_X^- that are not in S. These unnecessary additions do not invalidate the result but may correspond to redundant geometric calculations unless the representation scheme presented below is used.

5.3. Mixed-dimensional geometric model

To represent a solid part together with its additive and subtractive volume features, one needs to extend the boundary representation scheme and to provide support for representing decompositions of solids and of their complements in terms of boundaries of potentially overlapping features (Fig. 16). Furthermore, the construction of closing faces for surface features may

† See Property 13 in [27].

‡ See Property 12 in [27].

† "Classifying a point against a set consists of determining whether the point lies inside or outside of the set [38]."

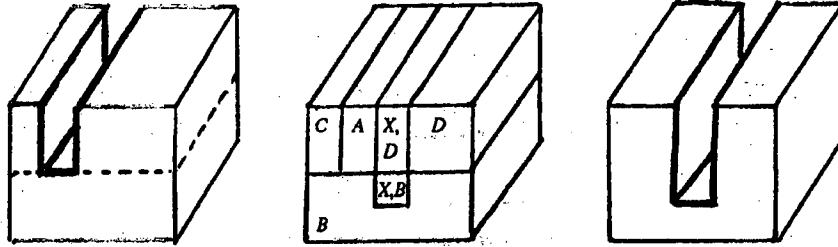


Fig. 23. Space decomposition for editing a surface feature: The slot surface feature in the solid $B \cup C \cup D$ (left) may be modified by creating the corresponding volume feature A and its modified version X , by constructing an SGC representation of the resulting space subdivision (center) and by setting the status of cells of $A \rightarrow X$ to IN, and of cells of $X \rightarrow A$ to OUT (right).

involve splitting faces of the part model and inserting internal faces in the part or creating external faces in its complement. Objects with internal structures and external dangling edges and faces are not supported by conventional solid modellers, and thus richer schemes for geometric representations must be used.

Several data structures that support representations of unions of quasi-disjoint subvolumes have been proposed (see [28–30, 42] for examples). Such schemes may be used to decompose a volume feature B into its active and inactive portions with respect to various Boolean combinations of other features. The Selective Geometric Complex (abbreviated SGC) data structure permits representation of such decompositions together with the modelled part, whether the feature has been added or subtracted [29].

SGCs provide a common framework for representing objects of mixed dimensionality, possibly with internal structures and incomplete boundaries. SGCs are composed of finite collections of mutually disjoint cells, which are open connected subsets of n -dimensional manifolds and generalize the concepts of edges, faces, and vertices used in most solid modellers. The connectivity between such cells is captured in a very simple incidence graph, whose links indicate “is-a-boundary-of” relations between cells. By setting the status (attribute) of certain cells to IN and others to OUT, one can associate various pointsets with a single collection of cells. When a cell’s status is IN, the cell is called “active,” and the pointset spanned by the cell is considered as part of the object; otherwise it is considered as part of the complement of the object. Consequently, the pointset of an SGC object needs not be homogeneous in dimension, nor even be closed or bounded. To support useful operations on SGCs. Boolean and other set-theoretical operations (closure, interior, boundary) have been decomposed into combinations of three fundamental steps for which dimension-independent algorithms have been developed [29]:

- a subdivision step, which makes two objects “compatible” by subdividing the cells of each object at their intersections with cells of the other object;
- a selection step, which defines “active” cells, i.e., cells whose status is IN; and
- a simplification step, which, by deleting or merging certain cells, reduces the complexity of an object’s

representation without changing the represented pointset and without destroying decompositions that are marked as important for applications.

Furthermore, combinations of these steps may produce a variety of special-purpose operations whose effect is controlled by simple predicates, or filters, for cell selection.

The subdivision step may be used to create a space decomposition that reflects the geometry of the part and of all its volume and surface features. Then, feature modifications may be performed by modifying the status of cells that belong to the appropriate volume features and the associated I or U zones. Fig. 23 illustrates how the result of a sequence of operations may be modified by selecting a surface feature and modifying it by replacing the volume feature A with X .

5.3.1. Application to model updating. A space decomposition technique may be used to split $A \oplus X$ into connected cells that are entirely inside or entirely outside of both I_A^S and U_A^S . Adding these cells to S or subtracting them from S only requires setting their status to IN or OUT. If such an approach is used, the main cost lies in the insertion of X in the space decomposition and in the classification of each cell of $A \oplus X$. The classification may simply be obtained by evaluating a Boolean expression. Using I_A^S or U_A^S instead of the expression for Z_A^S considerably reduces the cost of updating the model to reflect a feature modification (see Fig. 24).

5.3.2. Volume feature deletion. A particular case of editing is deletion. A volume feature could be deleted by altering a specification stored in a procedural model and by reexecuting the procedural model to create a new geometric model without the undesirable feature. The same result could be obtained directly by changing the status of all cells that lie in the intersection of that volume feature with its active zone.

For example, to delete a feature A in $(B \cup A) - C$, it suffices to change the status of cells in $A - (B \cup C)$, the active portion of A (Fig. 25). These changes can be done by traversing all the cells in A and classifying

[†] We assume that each feature has a reference to all the cells that it includes or that these cells may be efficiently identified in the SGC. If no provision is made for such a direct access, all cells of specific dimensions may have to be traversed to see if they belong to the appropriate feature.

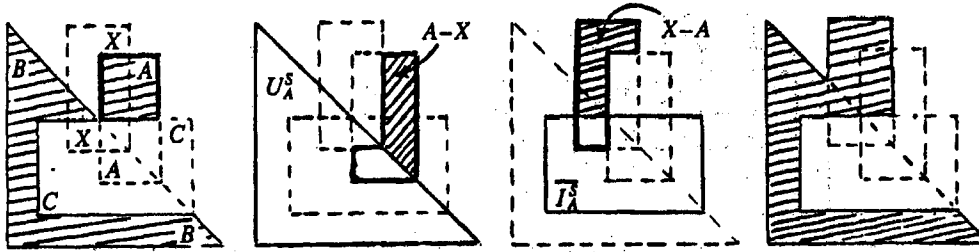


Fig. 24. Active zone in feature modification: S defined as $(B \cup A) - C$ is shown (left) superimposed on the space decomposition obtained by using the boundaries of the primitives A , B , C , and X . The primitive A is a misplaced additive volume feature in S and must be replaced with a similar feature X slightly shifted to the left and upwards. To obtain the correct object, first cells that belong to $A - X$ are classified against U_A^S (here $U_A^S = B$). The status of cells of $A - X$ outside of U_A^S is set to OUT. Note that one of these cells was already OUT (center left). Then, cells of $X - A$ are classified against I_A^S (here $I_A^S = C$). The status of the cells of $X - A$ inside I_A^S is set to IN (center right). Note that one of these cells was already IN. The result (cells whose attributes are IN) is shown (right).

each one against $B \cup C$. Since each cell contains (in its history) a list of the features to which it belongs, this classification amounts simply to evaluating a Boolean expression.

As discussed earlier, instead of changing the status of only the cells of A that lie in its active zone, one could simply set to OUT the status of cells of A that lie in its U -zone when A is an additive volume feature, or set to IN the status of cells of A that lie in its I -zone when A is a subtractive volume feature.

For example, in the solid $(B \cup A) - C$ (Fig. 25), the U -zone, $U_A^S = B$, is simpler than the active zone, $Z_A^S = B \cup C$, and thus a saving in classification time is achieved. This simplistic example does not reflect the importance of such savings, and one should consider that classification has to be performed for a large number of cells and that it may involve large Boolean expressions for both U_A^S and I_A^S .

6. CORRECTIVE VOLUMES

If a CSG tree representing the model is available, surface features could in principle be corrected by editing the tree. Suppose that no single node of the tree represents the appropriate volume feature. Each face of the feature may still be associated with one or more half-spaces of the CSG tree, so that modifying the position or shape of these half-spaces will affect the face, and thus the feature.

Editing half-spaces directly in the original CSG tree has the following drawbacks:

1. It may be difficult to recognize which half-space should be edited. Typically the boundary of more than one half-space coincides with each face, and techniques proposed in [27] for classifying half-spaces against their active zones (to predict which half-spaces will affect which portion of a particular face) may not be sufficient.
2. Editing half-spaces in a CSG tree to alter a particular portion of space will often produce undesired side-effects in other locations.
3. Editing half-spaces and not solid primitives can produce unbounded subsolids and makes it very difficult to implement popular performance-improving techniques for CSG algorithms that use bounding boxes around CSG primitives.
4. Designers' intentions expressed in a posteriori alterations of half-spaces are difficult to specify and to capture in the sequence of a procedural model. Consequently, the edited specification is not well suited for reparameterization or reuse in a different context. It is not even suited for further editing that involves a reevaluation of the sequence.

Reorganizing the CSG tree to regroup the relevant half-spaces of the feature into a single node that can be edited as a feature is not always possible since the result of evaluating a general Boolean expression is order-dependent. Consequently, this section proposes to use Boolean operations to edit surface features (the previous section dealt with volume features). These

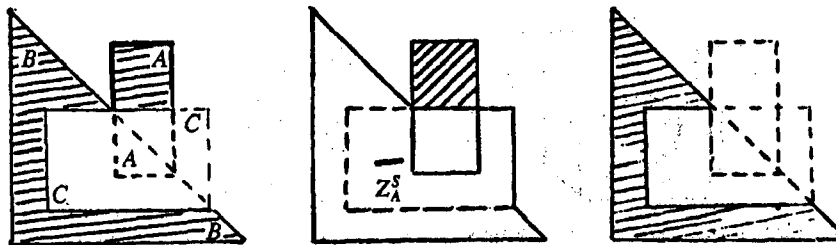


Fig. 25. Deletion using the Active Zone: Given the set S defined by $(B \cup A) - C$ (left) the additive volume feature A may be deleted by changing the active attribute of all the cells that lie in the active portion of A (center), which is $A - (B \cup C)$. The result is shown (right).

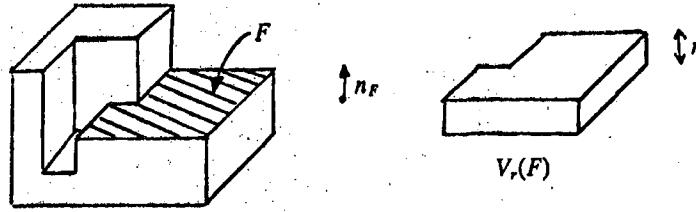


Fig. 26. Face extrusions: Extruding a planar base face (left) produces a volume (right) that is bounded by the base face and its offset by r , and by side faces obtained by sweeping the edges of the base face.

operations involve corrective volumes which are added to—or subtracted from—a solid model to alter a particular feature. These corrective volumes could be derived from volume features themselves derived from surface features by generating closing faces, but, as pointed out in Section 2, the derivation of closing faces remains a research issue. Furthermore, once a volume feature A is derived from the surface feature, it is not easy to produce a modified volume feature X so that $A - X$ and $X - A$ may be used as corrective volumes and added to—or subtracted from—the model to alter the surface feature without side-effects. Instead, the author proposes to construct corrective volumes by extruding appropriate faces of the surface feature. Examples of these extrusions are provided in the following part of this section.

Once corrective volumes are computed, whether through closing faces and volume features or through face extrusions, they must often be trimmed to avoid undesirable side-effects before they can be combined with the solid. Automatically computing the correct trimming expression is impossible unless the expression “undesirable side-effect” is formally defined. Indeed, the correctness of a feature editing operation depends on the function of the edited feature and on the function of its geometric relation with other features.

6.1. Extrusion of faces

A simplest corrective volume may be obtained by extruding a planar face called the base face along the normal to its supporting plane (see Fig. 26 for an example). Such a corrective volume may often be adequate to change the width or depth of features that have orthogonally oriented planar faces.

The extrusion $V_r(F)$ of a planar face F by a distance r is a volume formally defined by

$$V_r(F) = \{ \mathbf{p} + t\mathbf{n}_F \text{ with } 0 \leq t \leq r \text{ and } \mathbf{p} \in F \},$$

where \mathbf{n}_F is the normal unit vector to the surface containing F .

For faces on curved surfaces, a normal extrusion will be used (Fig. 27). It is simple extension of the above extrusion. The normal extrusion $V_r(F)$ of a curved base face F by a distance r is a volume formally defined as:

$$V_r(F) = \{ \mathbf{p} + t\mathbf{n}_F(\mathbf{p}) \text{ with } 0 \leq t \leq r \text{ and } \mathbf{p} \in iF \},$$

where $\mathbf{n}_F(\mathbf{p})$ is the normal unit vector to F at point \mathbf{p} and where iF is the relative interior[†] of F with respect to its supporting surface. The orientation of the normal is chosen in a consistent manner throughout F . Note that, for the semialgebraic surfaces popular in solid modelling, $\mathbf{n}_F(\mathbf{p})$ is well defined for the smooth portions of F (the relative interior of the base face), but needs not be well defined for the bounding edges of F , its cusps, or singularities.[‡] Therefore, $V_r(F)$ is obtained by extruding a nonclosed face and thus does not necessarily contain all its boundary; it needs to be regularized. $\mathbf{n}_F(\mathbf{p})$ can often be computed from the cross product of partial derivatives when F is defined in parametric form, or from a gradient when F is defined by an implicit equation. Such extrusion volumes are very simple to obtain for natural surfaces (see [43]).

Applications of extrusions and normal extrusions to feature modification are illustrated in Figs. 28 and 29, where the extrusions are used as corrective volumes and added to the part to modify a feature.

Such applications are clearly limited to simple cases where, for example, the extruded base face and the abutting faces meet at a right angle. Fig. 30 shows a



Fig. 27. Normal extrusion for curved base faces: A normal extrusion of a curved face that lies on a cylinder (left) is a volume (right) bounded by the original face, its offset by r , and by side faces that are subsets of ruled surfaces sustained by the edges of the original face. Note that the side faces are in the closure of $V_r(F)$ but are not in $V_r(F)$.

[†] The interior of a face is the face minus its bounding edges, cusps, or singular points.

[‡] The normal to a cone is not defined at its apex.

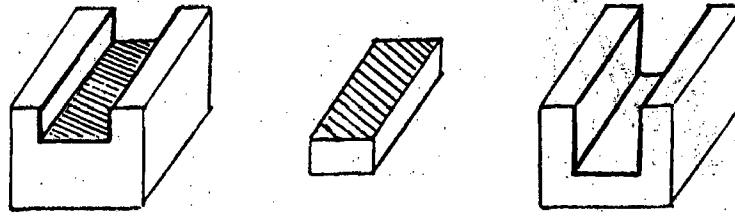


Fig. 28. Application of extrusion: To change the depth of the slot feature (left), a corrective volume (center) may be generated by extruding the floor of the slot and subtracted from the part (right).

counterexample for which a corrective volume generated by extruding a base face is inadequate.

To circumvent such limitations, an extended corrective volume may be generated and then trimmed using the abutting faces (see Fig. 31).

The extended corrective volume may be obtained by extruding an extension of the base face. A good candidate to use as base face extension is the entire surface that contains the base face, but in certain cases, to avoid side-effects in distant areas, it may be preferable to consider only a simple subset of that surface. This technique has been proposed by Requicha and the author in [12] for generating local fillets and blends by (1) growing and shrinking Boolean combinations of appropriate half-spaces, by (2) subtracting the result from the original combination to obtain an extended blend, by (3) trimming the blend to the desired shape near its ends, and by (4) adding the resulting corrective volume to the part or subtracting it.

Extended corrective volumes derived from a single base face are convenient for modifying a single dimension of a simpler feature (e.g., the width of a slot or the radius of a hole). In general, however a corrective volume involves more than one base face (for example, when the depth of a pocket with an uneven floor is to be changed). For such cases, instead of combining several corrective volumes, a single extended corrective volume may be obtained by extruding several faces along a common direction (Fig. 32). Such a generalized extrusion, $V_u(F)$ of a set of faces F along a direction u , may be formally defined as the Minkowski sum [44] of F with the line segment joining the origin O with the point $O + u$. And thus:

$$V_u(F) = \{p + tu, \text{ with } p \in F \text{ for } 0 \leq t \leq r\}.$$

6.2. Automatic derivation of a default trimming expression

Picking the base face and providing an offset distance does not in general provide an unambiguous specification for the corrective volume. It is thus necessary to provide facilities for automatically creating supersets of the desired corrective volumes and trimming expressions that produce the desired subsets. Trimming operations may involve nontrivial Boolean combinations of the half-spaces bounded by the abutting faces. This Boolean combination may, in principle, be derived from the entire CSG tree, if available, but this derivation may be difficult and will often require human intervention. A possible approach to assist the designer and suggest a default trimming CSG expression is to consider only half-spaces bounded by the faces of the solid that share edges with the base face and to eliminate other half-spaces from the tree. The correct Boolean combination of these half-spaces may be hard to derive, and the results may still be incorrect (see Fig. 33). The designer may have to provide an auxiliary trimming expression, but the system should be able to suggest a good default trimming expression.

Clearly, corrective volumes should be confined to a region where they do not destroy the effects of other features. For example, they may be trimmed by some other (but not necessarily all other) volume features.

Unfortunately, a surface feature, F , typically does not correspond to any individual node in the CSG tree and therefore is not associated with an active zone; the results derived in Section 5 may not be directly applied here.

To provide an often reasonable default trimming expression, the concept of a virtual active zone may be used. The virtual active zone of a surface feature F is defined as the active zone of the lowest node T in the

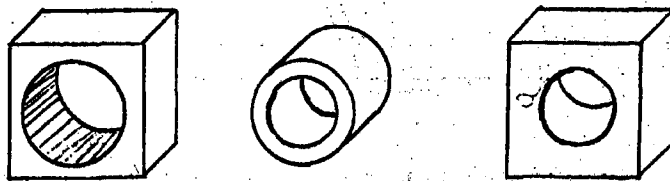


Fig. 29. Application of normal extrusion: To change the radius of a cylindrical hole (left), a corrective volume (center) may be generated through a normal extrusion of the cylindrical base face and added to the part (right).

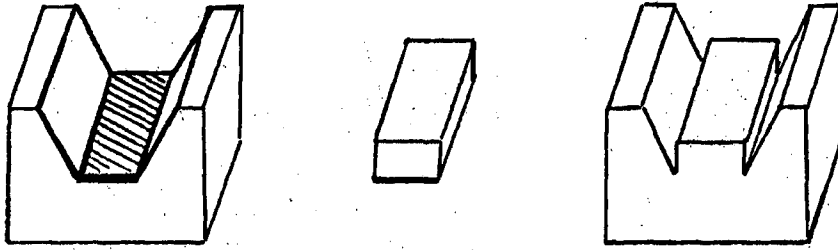


Fig. 30. Limitation of face extrusion: The geometric feature (left) has been edited by adding to the solid the corrective volume generated by extruding one of its faces (center). The result (right) is not correct.

CSG tree of S at which the surface feature F appears. This technique may be automated, but is only effective if at T one could delete and create again the corresponding volume feature without changing the final result. In other cases, fine tuning "by hand" the automatically generated virtual active zone may be necessary.

7. VALIDITY TESTS

Previous sections dealt with techniques for assisting the designer in performing object modifications using a feature-oriented syntax. It has been assumed that no automated solution exists and that human intervention is necessary to correct the side-effect of these editing operations. To further assist the designer, the system should support facilities for interrogating important properties of features. We shall refer to these properties using the global term of validity.

The validity of a feature or of a compound feature greatly depends on the nature of the feature and on the function played by the feature in a particular application. Addressed here are only the qualitative (or discrete) validity issues that can be expressed in terms of the presence or absence of specific cells in SGC structures. (Many other quantitative criteria may be easily addressed through geometric measures, derived from features, as described in [4].) Cells of interest have a specified dimension and belong to the appropriate combination of features. Note that this approach is not based on matching subsets of the adjacency graph of a boundary representation, but on the interrogation

of a rich data structure that captures various geometric entities and their relations.

Filters for cell selection in SGCs may be used for validity testing. The next section introduces the query operators, which provide the vocabulary necessary for expressing the filters. The subsequent section demonstrates their application on a few simple examples.

7.1. Interrogation operators for SGCs

Filters will be expressed in terms of the following queries that can be made to an SGC, O , or to a particular cell, C , of O . A list of typical filters, expressed as methods of cell or SGC objects, follows.

- $O.cells(k)$ returns the collection of cells of dimension k in O .
- $O.skeleton(k)$ returns the collection of cells of dimension less or equal to k in O .
- $C.boundary$ returns the collection of cells that bound C .
- $C.star$ returns the collection of cells bounded by C .
- $C.dimension$ returns the dimension of C .
- Given a cell D of $C.star$ such that $D.dimension = C.dimension + 1$, $C.nbhd(D)$ returns *leftdir*, *rightdir*, or *bothdirs*. The value *leftdir* means that, given the definition of "left" and "right" with respect to an orientation of C in the manifold containing D as an open subset, D lies on the "left" of C . Similarly, when $C.nbhd(D)$ returns *rightdir*, C bounds D on the "right." *bothdirs* means that C is an interior boundary "surrounded" by D , or more precisely that

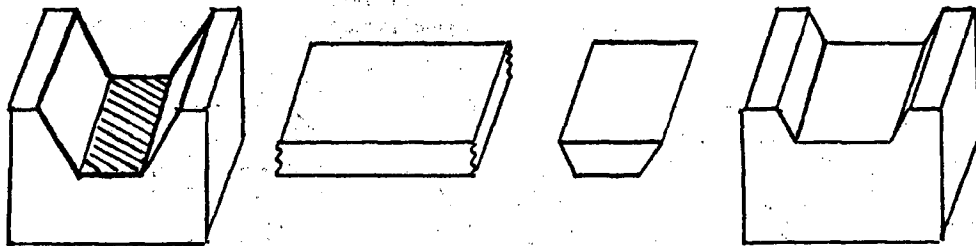


Fig. 31. Extended corrective volumes: To raise the floor of the slot (left) without modifying the position of its walls, an extended corrective volume is generated and trimmed (center). The result is added to the part (right).

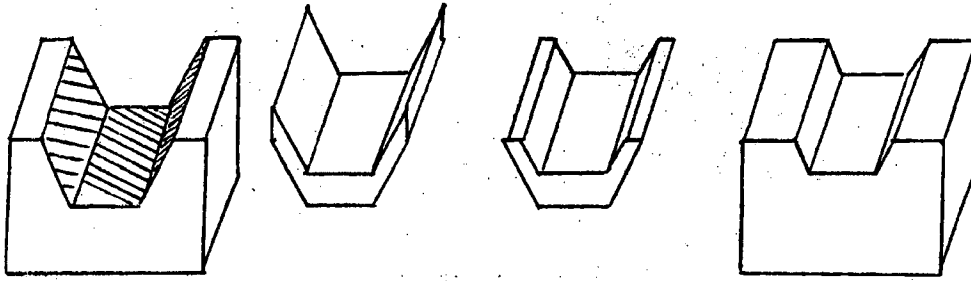


Fig. 32. Corrective swept volumes: To raise a slot in the model (left) without changing the shape of its floor, a corrective volume is obtained by sweeping upwards all the faces of the slot (center left), and then by trimming the result (center right) and adding it to the model (right). The dimensions of the floor remain unchanged and therefore the swept volumes of the different faces have different thickness.

C is contained in the topological interior (with respect to the manifold containing D) of the topological closure of D.

- C.history returns the set of features to which C belongs.
- Finally, C.status returns IN or OUT depending whether the cell is considered as actively contributing to the pointset of O or not.

For more formal definitions of these operators, the reader should consult [29].

7.2. Examples of application

Validity criteria are domain dependent, and the goal of this paper is not to derive them but to illustrate a technique for expressing them. Three simple examples involving a block of material and two volume features will be used. Simple tests that characterize each situation are proposed. These tests do not involve any geometric calculations, but simply search the SGC structure for cells that satisfy appropriate selection criteria. In fact, to improve the interrogation performance, references to cells of the SGC could be organized in a data base and accessed using their history, dimension, or other characteristics by standard data base queries.

The local inaccessibility from the top of a slot volume feature A in a part B may be detected by checking whether the "roof" face, F1, of A is connected on the outside (with respect to A) to a full-dimensional cell of B (Fig. 34). The test may be performed by selecting

cells D of F1.star that contains B in their history and such that $F1.nbhd(D) = \text{leftdir}$. (For simplicity, we assume that F1 is oriented so that the leftdir direction points toward the outside of A.)

Of course, the foregoing filter is not adequate for global accessibility, which may require performing Boolean operations on auxiliary volumes, such as the volume swept by the portion of the feature that is visible from the direction from which the feature is to be accessed.

To find whether two slots A and C are adjacent along a common face or not (Fig. 35), it suffices to inquire whether a 2D cell, or face, F2 exists such that it bounds a 3D cell of A on one side and a 3D cell of C on the other side.

The search may be confined to the common 2D cells of the boundaries of A and C, which may be accessed directly if a directory of cells that belong to each feature is maintained.

To test whether a slot C is contained in a slot A (Fig. 36), or more generally whether A and C interfere, it suffices to query if there is a three-dimensional cell whose history contains both A and C as well as B. Again, the selection may be efficiently performed by standard data base queries on the directories of cells that belong to A, B, and C organized by dimension.

8. CONCLUSION

Interactive editing of CAD models may be simplified by the use of volume and surface features. Surface fea-

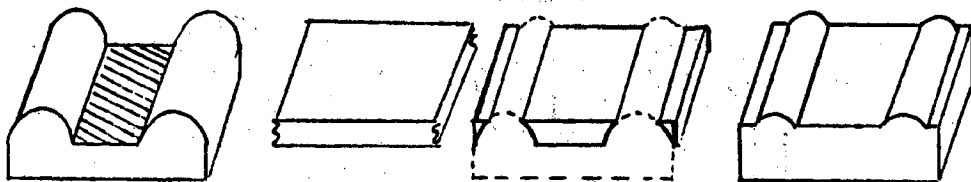


Fig. 33. Incorrect trimming CSG: To raise the floor of the slit in the model (left), an extended corrective volume is computed (center left). CSG expressions defined only in terms of half-spaces bounded by the faces of the original model are not adequate for trimming the corrective volume. An example of an incorrectly trimmed corrective volume is shown (center right), and the resulting model is shown (right).

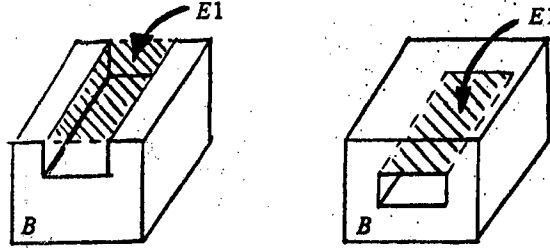


Fig. 34. Testing top accessibility: To distinguish between the correct positioning of the slot feature (left) and the incorrect one (right), one can simply inquire whether, in the corresponding SGC that represents the space decomposition, the roof face E1 of slot A is connected on its outside to full-dimensional cells of B.

tures are interactively selected by the user from existing faces of a model. Volume features may be created by addition or subtraction of material, or derived from surface features by defining the necessary closing faces. This paper addresses the issue of interrogating such volume and surface features as to their validity and their relation to each other and to the final part. It also proposes techniques for editing the model by modifying or deleting its features.

Interrogation techniques are based on a scheme for representing mixed-dimensional pointsets with internal structures. In that scheme, space, i.e., the modelled part and its complement, is subdivided into connected cells (volumes, faces, edges, and vertices) such that all points of any given cell belong to the same set of features. Feature validity and particular relations among features may be easily characterized by the existence or the absence of cells of specific dimensions associated with specific sets of features.

Model editing to alter or delete a volume feature may be performed by using a procedural representation of the designer's specification, locating in it the command that created a particular volume feature to be modified, editing this command, and reexecuting the entire procedural model. Reexecuting means computing the boundary of the geometric model, which may be an expensive process. When a CSG expression for the part is available, the reexecution may be limited to a particular domain defined by the intersection of the volume feature with its active zone. This technique

may be adapted to surface features by providing closing faces which define corresponding volume features and by considering the role these volume features play with respect to the CSG tree. On the other hand, surface features may be directly modified by constructing corrective volumes and by combining them to the part model through addition or subtraction. The use of extended geometric representations of sets with internal and external structures permits calculation of the effect of feature deletion without further geometric calculations.

Global access to the geometric elements of a particular feature is provided through intentional features, to which may be associated validity rules and methods for evaluating these rules or for measuring important properties of the feature. Intentional features may be created automatically when feature-based shape-modifying operations are used or by interactively selecting existing faces. Intentional features do not directly point to any geometric entity, but carry an unevaluated expression, constructed at feature creation or selection, which, when evaluated, returns appropriate geometric elements, if they exist. This indirect approach prevents inconsistencies between an abstract list of assumed features that could characterize some important aspects of a part and the actual presence and geometry of these features in the part. This point is essential if further shape modifications, done either by editing and replaying the designer's specification or by creating new features, can alter the geometry of a previously defined

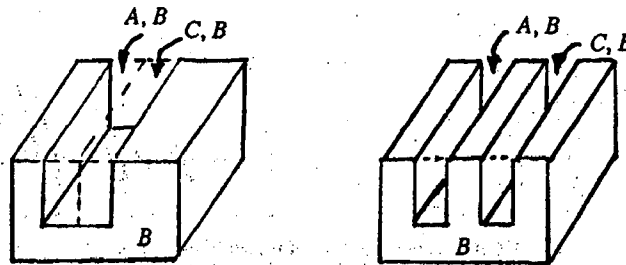


Fig. 35. Testing adjacency: For process planning, the two adjacent slots A and C (left) must be treated differently from the two disconnected slots (right).

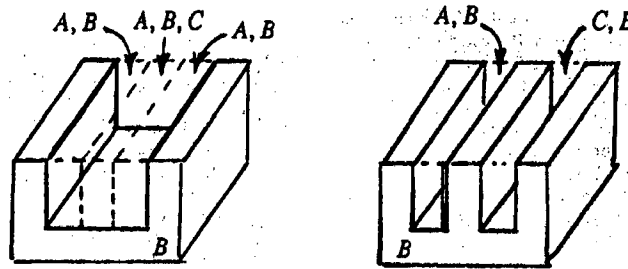


Fig. 36. Testing containment: Machining a slot C that lies inside a slot A may result in suboptimal manufacturing processes. Therefore, such a situation (left) must be distinguished from a normal situation where the two slots are disjoint (right).

feature to the point that it no longer exhibits the expected geometric characteristics.

Acknowledgments—The original motivation for a significant part of this work has been provided by Professor Michael Pratt's suggestion to use nonmanifold geometry for representing volume features and their interrelations. It became clear that the concepts and data structures for Selective Geometric Complexes, which have been designed at IBM by Michael O'Connor and the author, can be used to facilitate the explicit representation and interrogation of the geometric relations between features. The use of intentional features as a means for interrogating and editing a design has been investigated by Paul Borrel, Lee Nackman, and the author, as part of the development of the MAMOUR prototype system, and also by Franklin Gracer and the author, while attempting to incorporate "persistent" features in the GDP solid modeller. Furthermore, techniques for using intentional features the unevaluated references to boundary elements and for ensuring that these references remain meaningful, even when the specification has been edited, have been studied by Paul Borrel and the author. The idea of using trimmed corrective volumes to edit solids defined in CSG was proposed by Professor Aristides Requicha and the author at the University of Rochester. The original concepts of Active Zones in CSG were introduced by Professor Herbert Voelcker and the author at the University of Rochester. The author thus wishes to thank Michael Pratt for his inspiration and Paul Borrel, Franklin Gracer, Michael O'Connor, Aristides Requicha, and Herbert Voelcker for their contributions to certain aspects of this work. The author is also grateful to Vijay Srinivasan and Michael Wesley for supporting this work, to Professors Fahrad Arbab and Tetsuo Tomiyama for encouraging its publication, and to Erik Jansen, Martti Mäntylä, and several other colleagues and friends for their constructive comments.

REFERENCES

1. T. Tomiyama and H. Yoshikawa, Extended general design theory, In *Design Theory for CAD*, H. Yoshikawa and E. A. Warman (Eds.), North-Holland, The Netherlands, 95-130 (1987).
2. A. A. G. Requicha and S. C. Chan, Representation of geometric features, tolerances and attributes in solid modellers based on constructive geometry. *IEEE Journal of Robotics and Automation* 2(3), 156-166 (September 1986).
3. J. R. Dixon, J. C. Cunningham and M. K. Simmons, Research in designing with features, IFIP Working Group 5.2 Conference on Intelligent CAD, Boston, October 5-8, 1987.
4. J. R. Rossignac, P. Borrel and L. R. Nackman, Interactive design with sequences of parameterized transformations. *Proceedings of the Second Eurographics Workshop on Intelligent CAD Systems: Implementation Issues*, April 11-15, 1988, Veldhoven, The Netherlands, 97-127. Also available as IBM Research Report RC 13740, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, May 1988.
5. J. R. Rossignac, P. Borrel and L. R. Nackman, Procedural models for design and fabrication. *Proceedings of the MIT Sea Grant Symposium*, Cambridge, MA, October 24-26, 1988. Also available as IBM Research Report RC 14056, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, October 1988.
6. A. A. G. Requicha and H. B. Voelcker, Constructive solid geometry. Tech. Memo. No. 25, Production Automation Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
7. Y. T. Lee and A. A. G. Requicha, Algorithms for computing the volume and other integral properties of solids: I—Known methods and open issues, II—A family of algorithms based on representation conversion and cellular approximation. *Comm. ACM* 25(9), 635-641 (September 1982).
8. J. E. Bobrow, NC machine tool path generations from CSG part representations. *Computer Aided Design* 17(2), 69-76 (March 1985).
9. J. R. Rossignac and A. A. G. Requicha, Depth buffering display techniques for constructive solid geometry. *IEEE, Computer Graphics and Applications* 6(9), 29-39 (September 1986).
10. A. A. G. Requicha and H. B. Voelcker, Boolean operations in solid modelling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE* 73(1), 30-44 (January 1985).
11. K. J. Weiler, Edge-based data structure for solid modelling in curved surface environments. *IEEE Computer Graphics and Applications* 5(1), 21-40 (January 1985).
12. J. R. Rossignac and A. A. G. Requicha, Constant radius blending in Solid Modelling. *Computers in Mechanical Engineering* 3(1), 65-73 (July 1984).
13. M. J. Pratt, Synthesis of an optimal approach to form feature modelling. *ASME Computer and Engineering Conference*, San Francisco, California, August 1-3, 1988.
14. J. J. Shah, P. Sreevalsan, M. T. Rogers, R. Billo and A. Mathew, Current Status of Features Technology, Report R-88-GM-04.1, CAM-I Inc., Arlington, TX, November 1988.
15. T. Tomiyama and P. J. W. ten Hagen, Representing Knowledge in Two Distinct Descriptions, Extensional vs Intensional. Report CS-R8728, Center for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, June 1987.
16. A. A. G. Requicha, Mathematical models of rigid solid objects. Tech. Memo. No. 28, Production Automation

- Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
17. L. K. Kyprianou, Shape classification in computer aided design. Ph.D. dissertation, Christ's College, University of Cambridge, U.K., July 1980.
 18. A. A. G. Requicha, Representation of tolerances in solid modeling: Issues and alternative approaches. In *Solid Modelling by Computers*, M. S. Pickett and J. W. Boyse (Eds.), Plenum Press, New York, 3-22 (1984).
 19. M. J. Pratt and P. R. Wilson, Requirements for the support of Form Features in a Solid Modelling System. Report No. R-85-ASPP-01, CAM-I Inc., Arlington, TX, 1985.
 20. M. R. Henderson, Extraction of feature information from three dimensional CAD data. Ph.D. Dissertation, Purdue University, May 1984.
 21. J. R. Dixon and C. L. Dym, Artificial intelligence and geometric reasoning in manufacturing technology. *Applied Mechanics Reviews* 39(10), (October 1986).
 22. A. A. G. Requicha and J. Vandenbrande, Automatic process planning and part programming. Institute for Robotics and Intelligent Systems, Report IRIS 217, University of Southern California, Los Angeles, April 1987.
 23. V. C. Lin, D. C. Gossard, and R. A. Light, Variational geometry in computer aided design. *ACM Computer Graphics* 15(3), 171-177, (August 1981).
 24. A. P. Ambler and R. J. Poppelstone, Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence* 6, 157-174 (1975).
 25. D. Gossard, R. P. Zuffante and H. Sakurai, Representing dimensions, tolerances, and features in MCAE systems. *IEEE Computer Graphics and Applications* 8(2), 51-59 (March 1988).
 26. J. R. Rossignac, Constraints in Constructive Solid Geometry. Proceedings 1986 Workshop on Interactive 3D Graphics, University of North Carolina, Chapel Hill, NC 27514, F. Crow and S. M. Pizer, Eds., ACM Press, pp. 93-110, October 23-24, 1986. Also available as IBM Research Report RC 12356, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, September 1986.
 27. J. R. Rossignac and H. B. Voelcker, Active zones in CSG for accelerating boundary evaluations, redundancy elimination, interference detection, and shading algorithms. *ACM Transactions on Graphics* 8(1), 51-87 (January 1989).
 28. K. J. Weiler, The radial edge structure: A topological representation for non-manifold geometric modeling. In *Geometric Modeling for CAD Applications*, M. Wozny, H. McLaughlin and J. Encarnacao (Eds.), Springer-Verlag, 37-68, (May 1986).
 29. J. R. Rossignac and M. A. O'Connor, SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries, IBM Research Report RC14340, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, January 89. Also in "Geometric Modeling for Product Engineering," Proceedings of the 1988 IFIP/NSF Workshop on Geometric Modelling, Rensselaerville, NY, September 18-22, 1988. M. Wozny, J. Turner and K. Preiss, (Eds.), North-Holland, The Netherlands, 145-180 (1989).
 30. G. Vanecsek and D. Nau, Non-regular decomposition: An efficient approach for solving the polygon intersecting problem. Proc. Symposium on Integrated and Intelligent Manufacturing at the ASME Winter Annual Meeting, 271-279, 1987.
 31. R. B. Tilove, A null-object detection algorithm for Constructive Solid Geometry. *COMM ACM* 27(7), 684-694, (July 1984).
 32. L. R. Nackman, M. A. Lavin, R. H. Taylor, W. C. Dietrich and D. D. Grossman, AML/X: a programming language for design and manufacturing. Proceedings of the IEEE Fall Joint Computer Conference, Dallas, TX, pp. 145-159, November 2-6, 1986.
 33. A. A. G. Requicha and J. Vandenbrande, Form features for mechanical design and manufacturing. Report IRIS#244, Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782, October, 1988.
 34. S. D. Roth, Ray casting for modeling solids. *Computer Graphics and Image Processing* 18(2), 109-144 (February 1982).
 35. R. B. Tilove, A. A. G. Requicha and M. R. Hopkins, Efficient editing of solid models by exploiting structural and spatial locality. Tech. Memo. N. 46, Production Automation Project, Univ. of Rochester, May 1984. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
 36. A. R. Halbert, S. J. P. Todd, and J. R. Woodward, Generalizing active zones for set-theoretic solid models. *Computer Journal* (UK) 32(1), 86-89 (February 1989).
 37. R. B. Tilove, Exploiting spatial and structural locality in geometric modelling. Tech. Memo. No. 38, Production Automation Project, University of Rochester, October 1981. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
 38. R. B. Tilove, Set membership classification: A unified approach to geometric intersection problems. *IEEE Trans. on Computers* C-29(10), 874-883 (October 1980).
 39. J. R. Woodward and K. M. Quinlan, Reducing the effect of complexity on volume model evaluation. *Computer-Aided Design* 14(2), 89-95 (1982).
 40. J. R. Woodward, Eliminating redundant primitives from set-theoretic solid models by a consideration of constituents. *IEEE CG&A*, 38-47 (May 1988).
 41. S. A. Cameron and J. R. Rossignac, Relationship between S-bounds and Active Zones in Constructive Solid Geometry. IBM Research Report PC14246, T. J. Watson Research Center, Yorktown Heights, NY, December 1988. To appear in the proceedings of the International Conference on the Theory and Practice of Geometric Modelling, FRG, October 3-7, 1988.
 42. F. Arbab, Set models and Boolean operations for solids and assemblies. Technical Report CS-88-52, Computer Science Department, University of Southern California, Los Angeles, CA, 90089-0782, July 1985.
 43. J. R. Rossignac, Blending and Offsetting Solid Models. Tech. Memo. No. 54 (Ph.D. Dissertation), Production Automation Project, Univ. of Rochester, June 1985. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
 44. J. Serra, Image analysis and mathematical morphology. Academic Press, New York (1982).

Applicant : Somashekar Ramachandran
Subrahmanyam
Serial No. : 10/651,452
Filed : August 29, 2003
Page : 33 of 35

Attorney's Docket No.: 15786-018001

Exhibit B

Rossignac and Requicha, "Constructive Non-Regularized Geometry," *Computer-Aided Design*,
Vol. 23, No. 1, pp. 21-32 (1991).

Constructive Non-Regularized Geometry

Jarek R. Rossignac

Interactive Geometric Modeling
IBM, T.J. Watson Research Center
P.O. Box 704, Room J2-C03
Yorktown Heights, New York 10598
Phone: 914-784-7630, Fax: 914-784-7455
Email: jarek@ibm.com

Aristides A.G. Requicha

Programmable Automation Laboratory
University of Southern California
Los Angeles, California 90089-0782
Phone: 213-743-3805, Fax: 213-747-0820
Email: requicha@libari.usc.edu

ABSTRACT

Solid modeling is concerned with the construction and manipulation of unambiguous computer representations of solid objects. These representations permit to distinguish between the interior, the boundary, and the complement of a solid. They are conveniently specified in CSG (Constructive Solid Geometry) by a construction tree that has solid primitives as leaves and rigid body motions or regularized Boolean operations as internal nodes. Algorithms for classifying sets with respect to CSG trees and for evaluating the boundaries of the corresponding solids are known, at least for simple geometric domains. Emerging CAD applications require that we extend the domain of solid modelers to support more general and more structured geometric objects. This paper focuses on dimensionally non-homogeneous, non-closed pointsets with internal structures. These entities are well suited for dealing with mixed-dimensional ("non-manifold") objects in \mathbb{R}^n that have dangling or missing boundary elements, and that may be composed of several regions. A boundary representation for such objects has been described elsewhere. In this paper, we propose to specify and represent inhomogeneous objects in terms of Constructive Non-Regularized Geometry (CNRG) trees that extend the domain of CSG by supporting non-regularized primitive shapes as leaves and by providing more general set-theoretic and topological operators at interior nodes. We also provide filtering operations that construct CNRG objects from aggregates of selected regions of other CNRG objects. We present a syntax and semantics of the operators in CNRG and outline some basic algorithms for classifying pointsets with respect to the regions of objects represented by CNRG trees.

Keywords: Geometric Modeling, Solid Modeling, Constructive Solid Geometry, Inhomogeneous Objects, Non-Regularized Pointsets, Non-Manifold Topology, Internal Structures, Algorithms, Set Membership Classification.

1.0 INTRODUCTION

Solid modeling is concerned with unambiguous computer representations for physical solid objects, and with algorithms that process such representations. Earlier CAD systems represented solids by collections of edges, known as *wireframes*, which sometimes correspond to several solids—i.e., they are ambiguous or incomplete—or to no solid at all—i.e., they are invalid. Much of the initial impetus of solid modeling theory and technology was motivated by the desire to ensure the geometric completeness and validity of representations and the correctness of algorithms. Solids were modeled mathematically by *r*-sets, which are bounded, closed, regular, semi-analytic subsets of \mathbb{R}^3 [1]. Some authors considered a more restrictive modeling domain consisting only of *r*-sets whose boundaries are 2-manifolds [2,3]. (Informally, a 2-manifold is a collection of 2-D *faces*—for example, triangles—such that each edge is shared only by two faces and each vertex is the apex of only one *cone* of faces [4,5].) Although all solid modelers internally manipulate lower-dimensional geometric entities such as curves and surfaces, typically a user cannot access directly such entities, or use them to construct other non-solid objects. (Some of the earlier solid modelers such as BUILD [6] also accommodated 1-D and 2-D objects, called *shells* and *wires*, whereas others such as PADL-2 [7] provided procedural facilities for manipulating such entities only as part of a solid's boundary.)

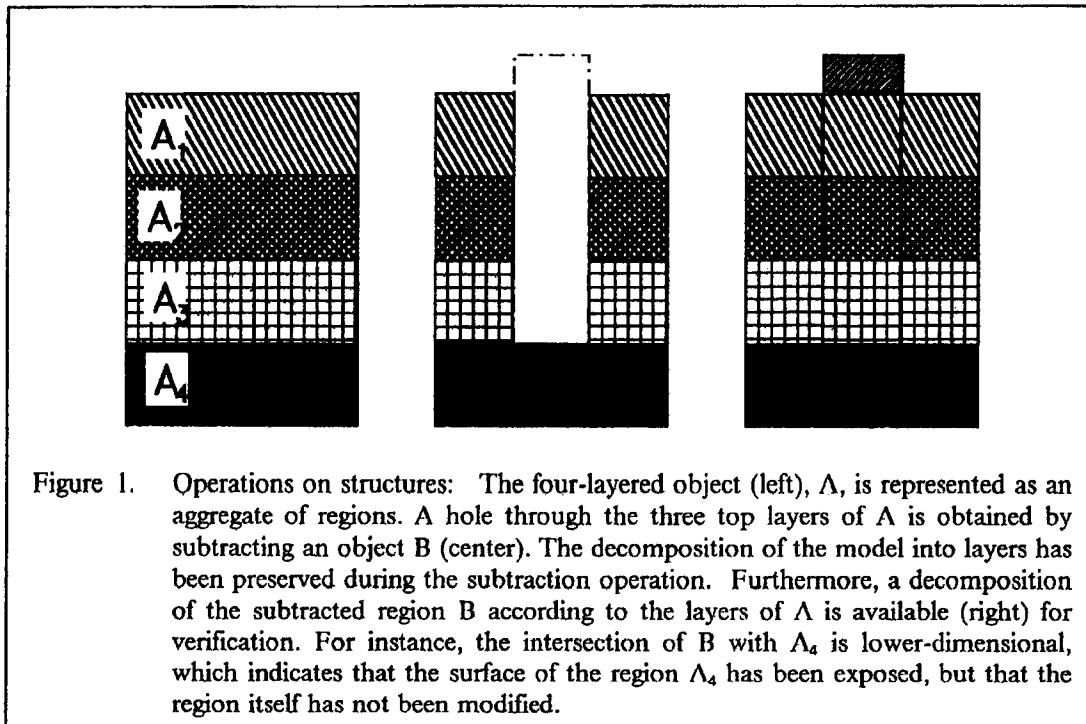
The most useful schemes for representing solids have been, and still are, Constructive Solid Geometry (CSG) and Boundary Representations (BReps) [1]. A CSG representation is a tree whose internal nodes represent either Boolean operations or rigid motions, and whose terminal nodes correspond to primitive solids such as blocks, cylinders, and tori. A BRep is a graph with nodes corresponding to the faces, edges and vertices that define a solid's topological boundary. Much has been said (correctly and incorrectly...) about the virtues of each of these representations. We, and many other geometric modeling practitioners, believe that both representations are useful and complementary. For example, a BRep is excellent for graphic user interaction, whereas CSG is very concise and easily supports parameterized object families. Current solid modelers typically contain both CSG and BReps in some form, and this trend is likely to continue indefinitely.

Many CAD/CAM applications now being developed require a geometric domain that extends beyond solids. The two following examples illustrate some of the requirements. Other examples may be found in [8,9,10,11] and range from feature-based design to stereolithography and finite-element methods.

Contact relationships are crucial in kinematics, assembly planning, grasping, and related areas. How can one tell if two solid objects A and B are in contact? What is their region of contact? Contact exists if the boundaries of the solids intersect and their interiors are disjoint. Alternatively, contact may be formalized by requiring that A and B intersect but that their regularized intersection be empty. (Recall that the regularized intersection is the topological closure of the interior of the standard intersection [1].) The region of contact is simply the intersection of the topological boundaries of A and B. This example shows that contact calculations involve regularized and non-regularized Boolean operations, as well as topological operations such as boundary and interior. Observe that regularized operations may be expressed in terms of their non-regularized counterparts plus closure and interior. Therefore, a modeler that supports the standard Boolean—intersection, union, difference and complement—and topological—interior, closure and boundary—operations on solids and lower-dimensional geometric entities is sufficiently powerful for studying contacts between objects.

Many practically-important objects such as composite-material aircraft parts and semiconductor circuits are composed of several regions with different material properties. Observe that the object in Figure 1 (left) cannot be represented in a solid modeler as the union of its regions, because the union operator obliterates the *internal boundaries* between the different regions. Furthermore, the subtraction operation performed in Figure 1 (center) preserves the structure of the object. This is not the case for Boolean operations commonly supported in solid modelers. Finally, the decomposition of the object's complement, Figure 1 (right), is available for validity checks. This example shows that geometric modelers should accommodate structured objects composed of several re-

gions, and that operators that manipulate simultaneously all the regions of a structured object are very useful.

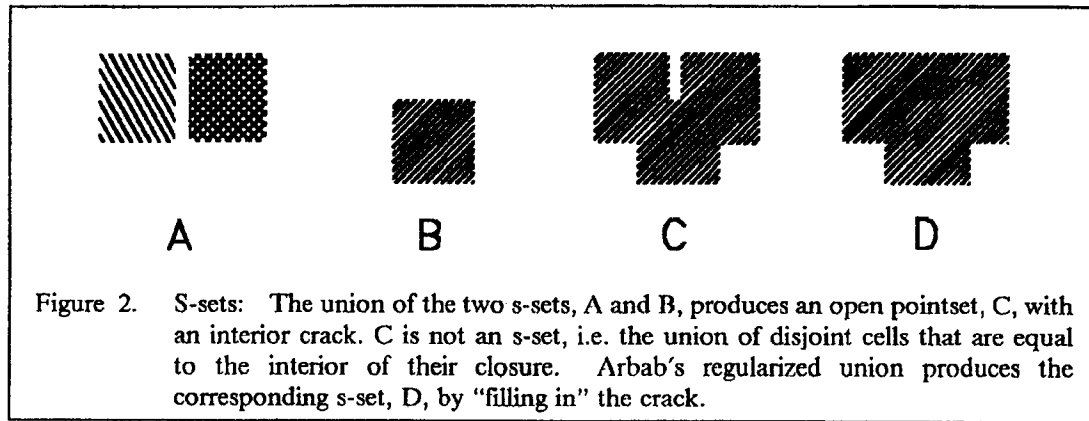


Several definitions have been proposed to extend the notion of solids to encompass more general geometric models. Many are based on mathematical definitions of simplicial complexes or related concepts which support pointsets that need not be dimensionally homogeneous. These extensions seek to represent, in the same model: a wireframe (i.e., a set of curves), a collection of surface elements (isolated faces), an aggregate of volumes, and even higher-dimensional sets.

Several boundary-based data-structures for representing "Non-Manifold Topologies" have been proposed in [12,13] for 2D objects, in [14,15,16,17,18] for 3D objects, and in [19,20,21] for higher-dimensional objects. These schemes are in general limited to the boundaries and internal structures of closed pointsets that are collections of vertices, lines segments, polygonal faces, polyhedra, and their higher-dimensional counterparts.

A representation based on "s-sets", i.e., finite aggregates of disjoint, open regularized (and thus full-dimensional) cells was proposed in [22]. These cells are equal to the interior of their topological closure. The set of such s-sets is closed under Boolean intersection, but not under the standard Boolean union. Indeed, the union of two s-sets needs not be an s-set (see Figure 2 and also Figure 8 of [22] for example). Consequently, the construction of s-sets models of assemblies may use intersection, difference, and "gluing" operations (i.e., operations that perform a union of cells whose interiors are disjoint), but fails when two objects with overlapping cells must be combined through a union operation. Arbab therefore defines a regularized version of the union operator, which returns the interior of the topological closure of the pointset produced by the union. Unfortunately, when applied to an aggregate of cells, this operator destroys all internal structures (i.e., removes faces that separate cells).

A more general scheme for representing arbitrarily-dimensioned non-closed pointsets that have internal structures and external "dangling" boundary elements was reported in [23], and is based on the concept of Selective Geometric Complexes (SGCs). An SGC object is an aggregate of mutually disjoint cells that are connected, dimensionally-homogeneous, relatively-open, singularity-free subsets of algebraic varieties. The vertices, edges, faces, and volume elements typically represented



in solid modellers, and their counterpart in higher dimensions, are cells. With each cell of an SGC is associated a flag, which specifies whether the cell is *active* or not. The pointset represented by an SGC is the union of (the pointsets of) its active cells. Furthermore, SGC cells may be tagged with attributes. The union of all active cells with the same attribute values forms a *region*. An SGC may therefore be decomposed into several regions, each region representing a dimensionally non-homogeneous pointset that need not be closed. High-level algorithms for performing Boolean and topological operations on SGC structures are briefly described in [23]. These algorithms maintain the decomposition of SGC objects into regions, while merging them through Boolean and other operations.

SGC representations are extensions of boundary graphs, and are produced by performing sequences of user-specified construction steps that transform or combine SGC representations of other objects. Editing commands, resulting geometric features, and arguments to algorithms that compute properties of the resulting structures are best specified in terms of individual *regions*, or their topologically-distinguishable subsets, which, for practical purposes, must be unambiguously defined in terms of the original users' input (for instance, references to construction steps or to primitive shapes) [24,25].

To compare the topological domains of geometric modellers one should examine the topological constraints imposed on the pointsets supported by these modellers and on their internal decompositions. Figure 3 shows characteristic examples of increasing topological domains. Only two-dimensional examples are used for clarity.

- A is an r-set with manifold boundary. (Each vertex is adjacent to exactly two edges.) Note that manifolds need not be composed of simply connected regions (one component of A has a hole), but the various connected components of manifolds must be totally disjoint.
- B is an r-set with non-manifold boundary. (A vertex is bounding more than two edges.) Note that regions whose interior are disjoint may share common vertices, but not common edges. Both A and B are equal to the closure of their interior.
- C is an s-set, (i.e. union of disjoint open regularized regions). C is composed of two disjoint regions that share a common boundary edge. This common edge is called "interior" because it is in the part of the boundary of C that lies in the interior of the closure of C. Obviously, closed sets do not have interior edges. S-sets cannot have non-separating interior boundary elements, such as interior vertices and cracks (i.e., interior boundary edges that do not separate two disjoint regions, but are connected to the same region on both sides).
- Non-separating interior boundary elements are possible with open sets, such as D, which do not contain their boundaries. Note that the boundary of D is non-manifold, i.e., there is no restriction on the number of edges bounded by any particular vertex.
- While the sets A through D are all homogeneously two-dimensional, E is not. It has a dangling edge that is not bounding any two-dimensional region.

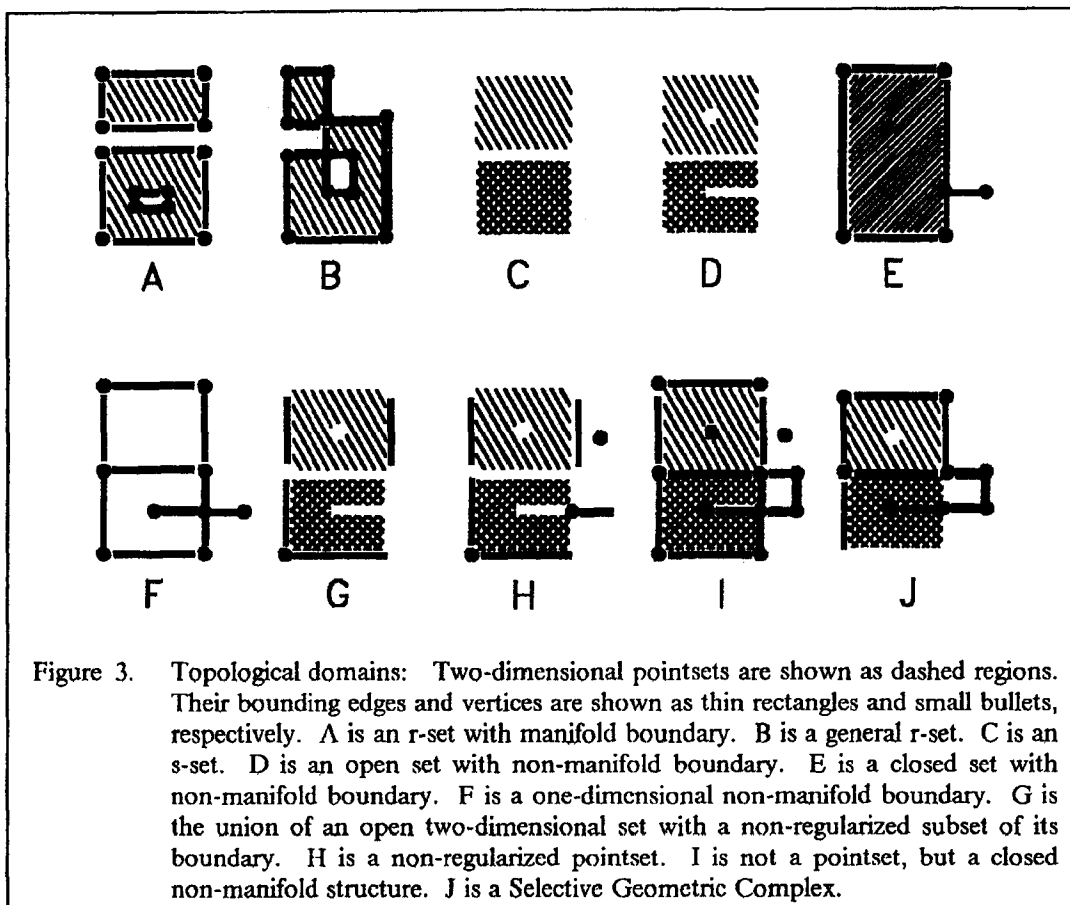
- F is a one-dimensional non-manifold boundary. Note that F has components that do not bound any finite two-dimensional region. Topologies, such as F , have been associated with the term “non-manifold” boundaries. Note that F implicitly defines two bounded two-dimensional regions, F_1 and F_2 . The term of “Non-Manifold Topology” has been used to popularize such topological domains in \mathbb{R}^3 [26], where sets like F would be considered as the *boundary* of a non-manifold set. However, the union of F with F_1 and F_2 is a closed set shown in E , whose boundary does not include all of F . Therefore, F cannot be the “boundary” of a two-dimensional pointset that would include these two regions.
- G is the union of an open two-dimensional set with a subset of its boundary, which is not a manifold (as imbedded in the two-dimensional space) and is not even a one-dimensional r-set (some edges are in F but their bounding vertices are not).
- H illustrates a general non-regularized pointset. It is a collection of disjoint connected open cells of dimensions 0, 1, and 2. This type of pointsets subsumes all previous types (A to G) illustrated in this figure. H exhibits cracks, interior vertices, and dangling edges and vertices that do not bound any region. Note however, that to produce a crack or a separating edge, the corresponding edge has been removed from the pointset, so that it can be included in the boundary.
- I is a closed non-manifold structure that subdivides the underlying pointset into two two-dimensional regions, having interior boundaries (filled cracks and points) that are distinguished from the rest of the two-dimensional regions because they are explicitly listed as boundary elements. Objects such as I are commonly associated with the category of “non-manifold topologies”, because the topological notion of a boundary has been relaxed to accommodate interior boundaries. Note that this type of decomposition is a generalization of simplicial complexes, as discussed in [23].
- J is an example of a Selective Geometric Complex [23], i.e., a collection of disjoint relatively open cells—here, two-dimensional regions, edges, and vertices. J is more general than I , because it needs not be closed. It is also more general than H , because it is a decomposition of space, and not the representation of a pointset and its boundary. Cells, as defined in [23], are required to be connected and dimensionally-homogeneous. Such cells correspond to a decomposition of pointsets that is too fine for practical interaction with users. In the present paper, we describe a coarser decomposition, where cells are grouped into non-homogeneous regions according to their origin in the design process.

In this paper, we propose a representation scheme that covers a rich set of inhomogeneous geometric objects and operations. We call it Constructive Non-Regularized Geometry (CNRG). CNRG trees represent objects that are aggregates (i.e., unions) of mutually disjoint regions. Each region is a pointset in \mathbb{R}^n and needs not be connected, regular, or even dimensionally homogeneous. The leaves of a CNRG tree correspond to parameterized primitive shapes such as volumes, faces, curve segments, or points. Internal nodes correspond to intermediate CNRG objects and are associated with topological and Boolean operations.

SGCs provide a model for representing non-regularized internally-segmented pointsets as a collection of connected open cells defined recursively in terms of their boundaries. CNRG trees yield an alternate representation for these pointsets as a collection of regions defined in terms of original primitive pointsets. Regions of CNRG objects need not be open nor connected and typically correspond to unions of SGC cells of various dimensions. CNRG models should be viewed as a primary model for user interaction because they support a high level vocabulary for expressing operations and regions, and because the CNRG trees are, similarly to CSG trees, easy to edit and archive. SGC models should be automatically derived from CNRG representations, as boundary representations are derived from CSG trees.

The proposed scheme extends CSG in several ways:

1. It uses standard (non-regularized) Boolean operations and topological operations—boundary, closure and interior. The regularized Boolean operations can be implemented in this scheme as three-operator sequences: standard Boolean, followed by interior and closure.



- Figure 3. Topological domains: Two-dimensional pointsets are shown as dashed regions. Their bounding edges and vertices are shown as thin rectangles and small bullets, respectively. A is an r-set with manifold boundary. B is a general r-set. C is an s-set. D is an open set with non-manifold boundary. E is a closed set with non-manifold boundary. F is a one-dimensional non-manifold boundary. G is the union of an open two-dimensional set with a non-regularized subset of its boundary. H is a non-regularized pointset. I is not a pointset, but a closed non-manifold structure. J is a Selective Geometric Complex.
2. It admits as primitives non-solid objects such as points, curves, or surfaces, and higher-dimensional objects.
 3. It introduces a new operator, called aggregation, which constructs structured objects composed of several regions. The aggregation operator is a formally-defined and more sophisticated version of the "assembly" operator provided by modelers such as PADL-2. Structured objects are not pointsets. They are collections of point sets, much like the cell complexes of algebraic topology [4]. Structured objects, also called in this paper CNRG objects, or simply objects, also have underlying pointsets, as cell complexes do. The underlying set of a CNRG object is the union of all the regions of the object, and therefore it is a pointset with no structure or "internal boundaries".
 4. It defines Boolean and topological operations on structured objects. These operations do not extend the domain of the scheme, since they can be expressed in terms of standard Booleans, topological operators, and aggregation. But they are very convenient because they manipulate simultaneously all the regions of their structured-object arguments. The definitions ensure that the underlying sets are treated consistently. For example, the union operator for structured objects produces another structured object whose underlying set is the union of the underlying sets of its arguments.

The remainder of this paper is organized as follows. Section 2 contains the formal definitions of the CNRG operators. Section 3 develops representations for the individual regions of structured objects. Section 4 discusses some of the fundamental algorithms for CNRG, and shows that they are very similar to their CSG counterparts. (A full-fledged suite of CNRG algorithms is still under development, and is outside the scope of this paper.) The final section summarizes the paper.

2.0 DEFINITIONS AND NOTATION

A CNRG object is a set of pairwise disjoint pointsets of \mathbb{R}^n called *regions*.

A region is a possibly non-regular (semi-analytic) subset of \mathbb{R}^n . Although a region of a CNRG object is a pointset and should be manipulated as a whole without further decomposition, for practical purposes it may be internally represented in some modellers as the union of primitive geometric elements, such as vertices, edges, faces, and other "cells" as defined in [23].

The pointset, pA , of a CNRG object, A , is the union of the pointsets of its regions.

Thus a CNRG representation not only defines a pointset, but also a decomposition (internal structure) of the pointset. Regions, and thus objects, need not be dimensionally-homogeneous, closed, or connected.

A CNRG tree is a rooted directed acyclic graph that represents a CNRG object. Leaves of the tree are CNRG primitives, which may be composed of more than one region. For simplicity, we assume that each primitive is given in its absolute position, although in practice, rigid body transformation nodes may be used. Internal nodes represent intermediate CNRG objects obtained by applying Boolean, topological, simplification, or filtering operators to the CNRG objects represented by their child-nodes.

To simplify the following discussion, we use " $|$ " to denote a "gluing" or aggregation binary operator that takes two disjoint regions, aggregates of regions (all of which are pairwise disjoint), or disjoint CNRG objects and produces a CNRG object that aggregates all the regions. Note that expressions involving only aggregate operators are order-independent.

In this paper, an uppercase letter denotes a CNRG object, and each of the object's regions is denoted by the same letter with a subscript. For example, A_1 and A_2 are two regions of the same CNRG object, A . If A is composed of only these two regions, we write, $A = \{A_1|A_2\}$. By definition, A_i and A_j are disjoint for $i \neq j$. A single-region object, A , is considered distinct from its region A_1 . We write $A = \{A_1\}$. Note that $A_1 = pA$ for single region objects.

In the following, we assume that S , A , B , C and D are CNRG objects. Furthermore, we often use C or S to denote the objects produced by applying to A (and to B) a Boolean or topological CNRG operator.

A and B are *disjoint*, if and only if the intersection of their pointsets, pA and pB , is empty.

Given two *regions*, A_i and B_j , $A_i \cap B_j$ denotes their intersection and $A_i \setminus B_j$ their difference in the standard set-theoretical sense. Note that $A_i \cap B_j$ and $A_i \setminus B_j$ are single regions that may be empty, disconnected, dimensionally inhomogeneous, and not closed.

A region is said to be *contained* in an object if it is contained in the pointset of the object. It need not correspond to the union of any subset of the object's regions.

We now define a set of new operators for CNRG objects. These operators are called: union, intersection, difference, complement, interior, closure, boundary, and regularization. They are analogues to the standard pointset operators. We use a different notation to distinguish between the two set of operators. For example, we use " \cap ", " \cup ", and " \setminus " for pointset intersection, union, and difference, and we use " \ast ", " $+$ ", and " $-$ ", for their CNRG counterparts. The topological complement of a pointset S is denoted \bar{S} . The corresponding CNRG operator will be denoted " c ".

The names of these operators were chosen according to their effect on the *pointsets* spanned by the objects and not on the internal decomposition of these pointsets. We have chosen to include the CNRG version of the standard topological operators (complement, interior, closure, and boundary) because these operators offer a well accepted and convenient syntax for specifying pointsets and for analyzing their topological properties. Our Boolean operators (union, intersection, and difference) correspond to the standard operators used in CSG. Except for the symmetric difference of A and B , which can be expressed as $(A - B) + (B - A)$, these three operators cover all non trivial bounded

Boolean combinations of two pointsets, A and B . Unbounded combinations may be expressed in terms of their bounded complements and of the complement operator.

The aggregate operator “|” and the pointset operator “ p ” were introduced in this paper to simplify the definitions. They need not be made available for use in CNRG trees. Instead, the union and simplification operators, precisely defined below, should be used.

A filter operator that eliminates selected regions will be introduced in the next section.

We define below the semantics for each of the CNRG operators.

2.1 Aggregation

Given n disjoint pointsets, A_i , the aggregation operation, denoted “|”, creates the corresponding CNRG object: $A = \{A_1 | A_2 | \dots | A_n\}$.

2.2 Simplification

The simplification, sA , of A is a CNRG object with a single region: the pointset pA . Thus, $sA = \{pA\}$.

2.3 Complement

The complement, cA , of A is an object composed of a single region that is the pointset complement of pA . Hence, $cA = \{\overline{pA}\}$.

2.4 Union

The union, $A + B$, is an aggregate of regions of the following three types: $A_i \cap B_j$, $A_i \setminus B$, and $B_j \setminus A$, for all combinations of regions, A_i , of A and regions, B_j , of B .

Potentially each region of A is split into two sets: the part in B and the part outside of B . The second set is a single region. The first set may be decomposed according to the decomposition of B into regions (see Figure 4 for example).

Union produces a subdivision of $(pA) \cup (pB)$ that is compatible with the decomposition of A and B into regions (see [23] for a more detailed discussion of subdivision). We call it “union” because the pointset $p(A + B)$ equals the set theoretic union, $pA \cup pB$.

We discuss in Section 3 a technique for distinguishing each region of $A + B$.

2.5 Intersection

The intersection, $A * B$, of A and B is the aggregate of all regions of the type $A_i \cap B_j$.

A region, A_i , of A is truncated to $A_i \cap pB$, and is subdivided according to the subdivision of B into regions. Consequently, $p(A * B) = (pA) \cap (pB)$, which justifies the name for this operator.

2.6 Difference

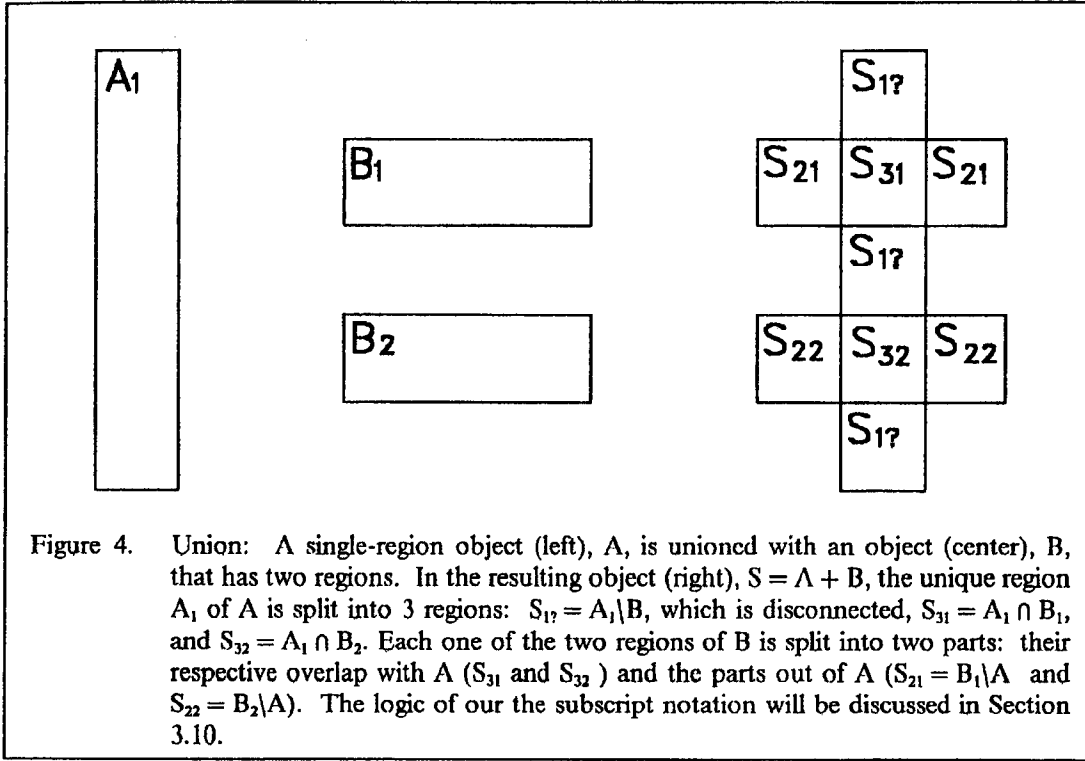
The difference, $A - B$, is the aggregate of all regions of the type $A_i \setminus pB$.

Clearly, $p(A - B) = (pA) \setminus (pB)$. It follows from the definition of union, intersection, and difference that $A + B = \{(A - B) | (A * B) | (B - A)\}$.

2.7 Interior

The topological interior, iA , of A in \mathbb{R}^n is the aggregate of regions, $A_i \cap \text{interior}(pA)$, that are the intersection of the regions of A with the topological interior of A in \mathbb{R}^n .

Note that, although iA is always full-dimensional, regions of iA need not be full-dimensional. Thus the *interior* operator returns a subset of the original object and preserves its internal decomposition into regions. The pointset, piA , of the interior of A equals the topological interior of pA .



2.8 Closure

The closure, kA , of A , is the aggregate composed of all the regions of A plus a single new region defined as the difference between the topological closure of pA and pA itself. Thus, we can write: $k\{A_1, \dots, A_n\} = \{A_1, \dots, A_n, |(\text{closure}(pA) \setminus pA)\}$, and pkA equals the topological closure of pA .

2.9 Boundary

The topological boundary, ∂A , of A is defined as: $\partial A = kA - iA$. The boundary operator does not necessarily return a single-region object. Note that $p(\partial A)$ equals the topological boundary of pA .

2.10 Regularization

The regularized version, rA , of A is defined as kiA . The pointset, prA , spanned by rA corresponds to a regularized solid as defined in [27] and equals rpA .

Note that regularization does not imply simplification (i.e., a regularized object may be composed of many non-regularized regions). However, taking the boundary, C , of a regularized object, A , will produce a lower-dimensional object whose pointset, pC , is the topological boundary of the pointset of the regularized object.

2.11 An example

The effect of interior, closure, and boundary are illustrated in Figure 5.

3.0 IDENTIFYING AND SELECTING REGIONS

A *CNRG expression* combines primitive regions through the operators defined above. A *CNRG tree* is a graph structure obtained by parsing the CNRG expression and generating an internal node for each operator and a leaf for each primitive object.

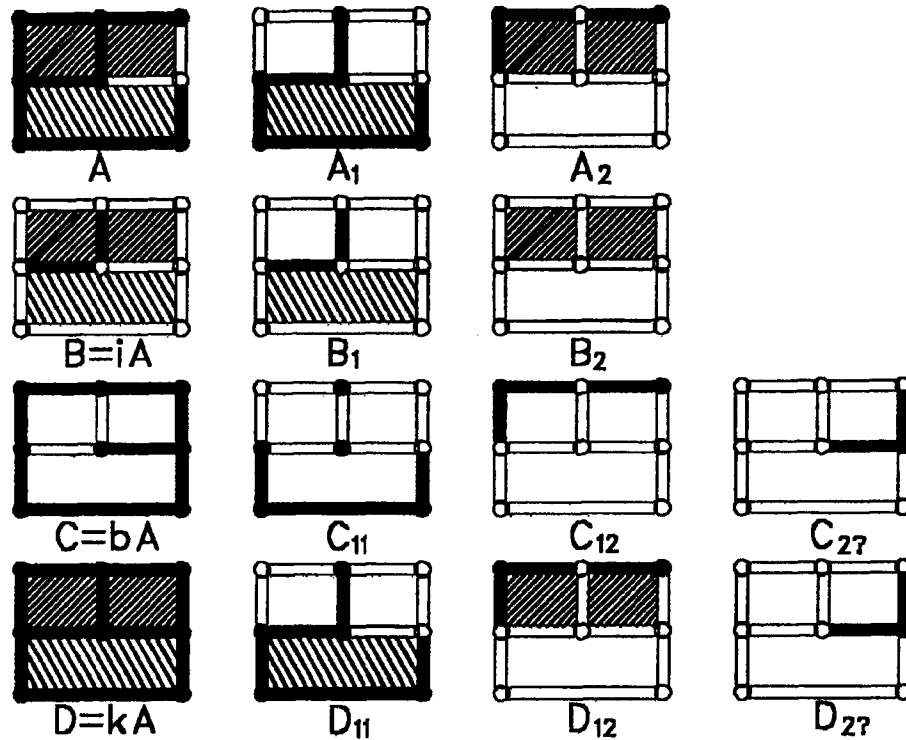


Figure 5. Operations: The top row shows a CNRG object, A , composed of two regions, A_1 and A_2 , shown on the same line. The next row shows B , the interior of A , and its two constituent regions. The third row shows C the boundary of A , (labelled "bA" in the figure) and its three constituent regions: a subset of A_1 , a subset of A_2 , and a new region, outside of A . The bottom row shows D , the closure of A , and its three constituent regions: A_1 , A_2 , and the same new region as in C . Note that $p(kA) = p(iA) \cup p(\partial A)$. The subscript notation, called the region's signature, and our convention for constructing it is introduced in section 3.10.

Union, intersection, difference, closure, interior, and boundary operations on objects, when followed by simplification, behave as their set theoretic counterparts applied to pointsets. However, without simplification, they return aggregates of pairwise disjoint regions. Therefore, when "evaluating" a CNRG tree or classifying a pointset with respect to a CNRG object, we need to take into account the object's decomposition into regions.

A CNRG tree defines an object, i.e., a pointset and its decomposition. But how can individual regions be identified? How can one test points for inclusion in a specific region? These questions will be addressed below.

First consider a simple example. Let A , B , C and D be single-region objects. The expression $(A + B) * (C + D)$ produces an object composed of possibly as many as 9 regions, some of which may be empty. We provide a set theoretic formulation for these 9 regions below. (For simplicity of notation, let the "+" operator have the lowest priority in CNRG expressions).

From the definition of union, $A + B = (A - B) \cup (A * B) \cup (B - A)$. Expanding the union in the object's expression yields

$$(A + B) * (C + D) = ((A - B) \cup (A * B) \cup (B - A)) * ((C - D) \cup (C * D) \cup (D - C)).$$

Distributing “*” over “|”, produces the following 9 regions:

$$\begin{aligned} (A + B)*(C + D) = & (A - B)*(C - D) \mid (A - B)*(C * D) \mid (A - B)*(D - C) \mid \\ & (A * B)*(C - D) \mid (A * B)*(C * D) \mid (A * B)*(D - C) \mid \\ & (B - A)*(C - D) \mid (B - A)*(C * D) \mid (B - A)*(D - C). \end{aligned}$$

For generality, we formulated each region in terms of operations on objects, but, because the primitive objects are single-region objects, each region has an equivalent set-theoretic expression. For example, $(A - B)*(C - D)$ is the single region: $(pA \setminus pB) \cap (pC \setminus pD)$.

How many regions does a CNRG have? The answer depends on the pointsets (it may have zero regions, because all the regions may be empty), but we can still enumerate all the possible regions and count their maximum number.

Suppose that a particular region of A (and a particular region of B for the case of binary operators) can be selected. Are these selections sufficient to unambiguously define any particular region of C? The answer depends on the operator. As shown below, for some operators, selecting a region of A (and a region of B) is insufficient to define a specific region of C. Unambiguous region selection may be achieved by associating with operators a *semantic interpretation*, which determines a specific group or “type” of regions produced by the operation. The possible interpretations are listed below for each operator.

Any region of C can be unambiguously defined by: (1) a region of the left child, A, of C, (2) a region of the right child, B, of C (if it has one), and (3) a semantic interpretation for the operator associated with C. If the primitives (the leaves of the CNRG tree) are single-region objects, then any particular region of the resulting CNRG object may be specified by selecting a semantic interpretation for the operators in the CNRG tree. Depending on the CNRG expression, some selections may not be necessary. For example, $s(A + B)$ has only one region, regardless of the interpretation of “+”.

In the following sub-sections we assume that A and B are composed of m and n regions respectively and that C corresponds to a single-operator expression involving a left child, A (and possibly a right child, B, for CNRG operators that take two arguments).

3.1 Simplification

The simplified version, sA , of A has only one region, and thus simplification has only one interpretation. No selection of a region of A is needed for specifying the region of sA .

3.2 Complement

The complement \bar{A} of A has also only one region. There is no need for selecting a region of A, nor an interpretation of complementation.

3.3 Union

The union $A + B$ has $(n + m + nm)$ regions of three different types. Each type corresponds to a different interpretation of “+”, which we distinguish by associating the corresponding subscript number with the “+” sign:

1. Interpretation $A +_1 B$ yields the m subsets of the m regions of A that lie out of pB .
2. Interpretation $A +_2 B$ yields the n subsets of the n regions of B that lie out of pA .
3. Interpretation $A +_3 B$ yields the (nm) regions of type $A_i \cap B_j$, that lie in both A and B.

Any region of $A + B$ may be selected by specifying the appropriate interpretation of “+” and by selecting a region of A and/or of B. (For example, when the first interpretation is used, the specification of a region in B is not necessary and will not change the result.)

Expressions involving union operators may define several regions. If these expressions combine single-region primitives, any particular region of the resulting CNRG object may be defined pre-

cisely by selecting the interpretation (i.e. type 1, 2, or 3, as described above) of some or all of the union operators.

3.4 Intersection

The intersection $A * B$ has (nm) regions of the form: $A_i \cap B_j$. Any one of these regions may be selected by specifying a region of A and a region of B . Intersection has only one interpretation.

Note that $A * B$ and $A +_3 B$ refer to the same thing, (i.e., identical pointset and identical decompositions). However, one uses "*" and "+" operators to specify pointsets and their decompositions. $A +_3 B$ is a convenient notation used in this paper to introduce a way of identifying regions of $A + B$. As discussed in Section 3.10, an aggregate of the subscripts of the operators of a CNRG expression may be used to identify a region or an aggregate of regions.

3.5 Difference

The difference, $A - B$, has m regions of the type $A_i \setminus pB$. Any one of these regions may be selected by specifying the corresponding region of A . There is only one interpretation for the difference operator and there is no need for specifying a region in B .

Note that $A - B$ and $A +_1 B$ define identical objects.

3.6 Interior

The interior, iA , of A has at most m regions that are subsets of the corresponding regions of A . Any one of these regions may be selected by specifying the corresponding region of A . There is thus only one interpretation for the interior operator.

3.7 Closure

The closure, kA , of A has $(m + 1)$ regions: the m original regions of A , plus one new region. Any one of the first m regions may be specified by selecting the corresponding region of A and by selecting the " k_1 " interpretation of closure. The last region is obtained by selecting the " k_2 " interpretation.

Note that $p(k_2A) = \text{closure}(pA) \setminus pA$. We have decided to consider all the pieces of k_2A as part of a single region, independently of their relations to regions of A , because some pieces of k_2A may simultaneously bound several regions of A (see for example D_{27} of Figure 5 on page 9). Note that the region k_2A needs not be full-dimensional or closed.

For example, given two single-region primitives, A and B , the expression $k(A + B)$ defines four regions specified as follows: $k_1(A +_1B)$, $k_1(A +_2B)$, $k_1(A +_3B)$, and $k_2(A + B)$, which is independent of the interpretation of "+".

3.8 Boundary

∂A has at most $(m + 1)$ regions. The m bounding regions that are subsets of regions of A correspond to the " ∂_1 " interpretation of " ∂ ". The only region not contained in A corresponds to the " ∂_2 " interpretation of " ∂ ".

Note that the operator interpretations " ∂_2 " and " k_2 " are equivalent.

The effects of interior, boundary, and closure, and the resulting regions are shown in Figure 5 on page 9.

3.9 Regularization

rA also has $(m + 1)$ regions. The m regions contained in A correspond to the " r_1 " interpretation, and the region outside of A corresponds to the " r_2 " interpretation of " r ".

3.10 Region signature and sub-objects

The subscript notation for the semantic interpretation may not be practical for users. It should either be hidden or replaced by a more intuitive notation in a modeler's user interface. But the notation is well suited for internal use in a modeller and has the advantage of providing a compact specification for a region of a CNRG object.

For example, let S be defined in terms of single-region primitives as $(A^*(B + C)) + D$. S has seven regions shown in Figure 6. Each one can be associated with a particular interpretation of the "+" operators. We define the region's *signature* as the concatenation of the interpretation numbers that correspond to the region, in the order in which their operators appear in the expression of the CNRG object. (We use numbers only for operators that have multiple interpretations.) We use the regions signature as a subscript of the object's name to define the corresponding region. For instance, $S_{11} = (A^*(B +_1C)) +_1D$.

Note the use of the wild card, "?", in Figure 6 when the semantic interpretation of a particular operator is irrelevant. For example, the region S_{22} does not depend on the interpretation of the first "+", since it lies outside of $p(A + B)$.

A region signature is complete when it unambiguously defines a single region, i.e. when all relevant semantic operators have been specified. An *incomplete signature* may be obtained by replacing by "?", in a complete signature, one or more semantic operator numbers. An incomplete signature is a filter that selects regions and defines a new CNRG *sub-object* decomposed into regions that correspond to all the possible combinations of the values that the "?" wild cards can take.

A sub-object is a valid CNRG object and may be used as such in CNRG expressions.

The wild card mechanism provides a simple notation for specifying sub-objects. For example, S_{71} in Figure 6 is composed of the three regions:

$$\{ (A^*(B +_1C)) +_1D \mid (A^*(B +_2C)) +_1D \mid (A^*(B +_3C)) +_1D \}.$$

For any sub-object, T , of a CNRG object, S , we have $pT \subset pS$. Furthermore, any region of T is a region of S . In particular $S_{71\dots7}$ is equivalent to S .

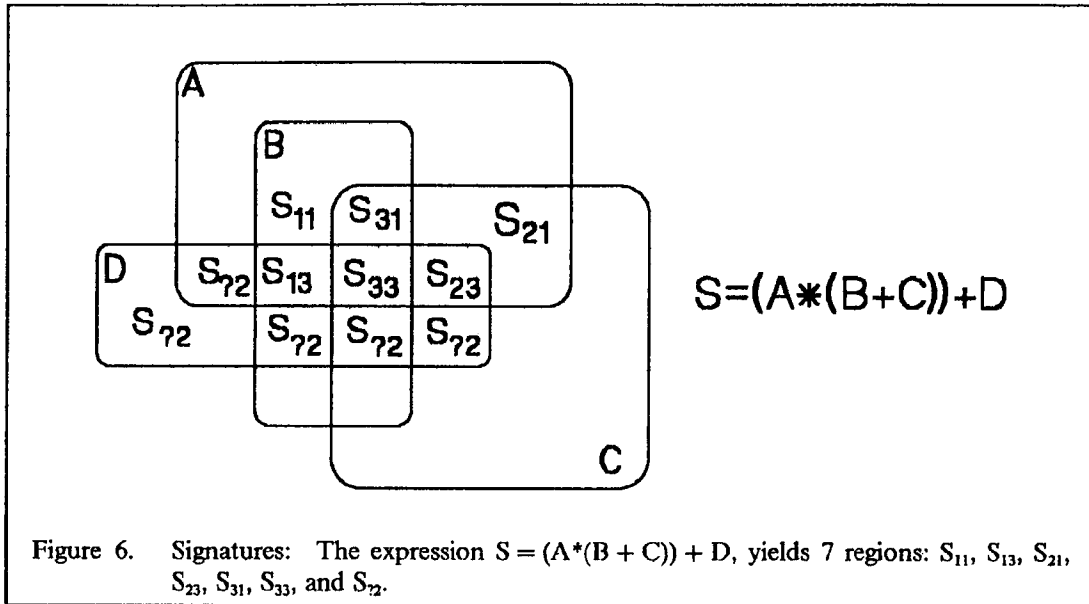
For a CNRG object defined in terms of multiple-regions primitives, a region signature concatenates (1) the numbers associated with the operators' semantic interpretation and (2) the numbers associated with the primitives' regions. Both types of numbers are sorted according to the ordered of the corresponding symbols in the object's expression. Figure 4 on page 8 shows another example of the use of subscripts for selecting regions of primitive objects.

A *filter* operator, $f_{\text{signature}}$, is associated with a valid *signature* of its CNRG object argument. For example, $f_{\text{signature}}A$ returns—as a CNRG object—the sub-object $A_{\text{signature}}$. Results of filter operations on the same object may be combined using the union operator to produce filters that cannot be expressed with a single signature of the object.

3.11 Constructive representation for regions

The CNRG operators are defined in Section 2 by specifying their effects on the *regions* of the arguments. All of the operators produce regions that can be expressed in terms of *standard* set theoretic and topological operations on the regions of the operands.

For example, the regions of $C = A - B$ are of form $C_i = A_i \setminus pB$ and $pB = \bigcup B_j$. Therefore, each C_i can be expanded in terms of A_i and B_j through standard set theoretic difference and union. If A and B are themselves composite CNRG objects, we can represent their regions constructively, and apply recursion until we reach the primitives. It follows that any region of a composite object can be represented constructively in terms of standard set-theoretic and topological operations on the regions of the primitives in the object's CNRG tree. The constructive representation for a region is easy to obtain from the CNRG tree and the interpretation attached to some of the operators.



The existence of constructive representations for regions is an important property of CNRG. Constructive representations for regions are similar to CSG and can be processed by similar algorithms. Whereas CSG algorithms deal only with *regularized* set operations on solids, CNRG algorithms must process non-solid regions through non-regularized set operations as well as topological operations. Thus, CSG algorithms cannot be used directly for CNRG, but must be suitably modified. Work on CNRG algorithms is under way. Preliminary results are reported in the next section.

3.12 Regions in the complement

For some applications, such as the analysis of the validity of design features [11], regions outside the pointset of the object (for instance slots removed from a part model) and their interactions may be of interest.

Consider the object $S = (A - B) - C$, of Figure 7, obtained by subtracting from the primitive A two slots, B and C. Although regions of $B + C$ are outside of S, they must be analyzed to derive the topological relation between the two slots. For example, the existence of an intersection $pA \cap pB \cap pC$ of B and C inside A (Figure 7 a and b) indicates that, in order to reduce manufacturing costs, both features should probably not be machined independently.

Most outside regions of interest are bounded, lie inside the union of the primitives' pointsets, and can be formulated as CNRG expressions that combine the original primitives.

In particular, replacing specific "-" and "*" operators of the CNRG tree with "+" operators produces a useful decomposition of the original object and of the relevant portion of its complement.

Let S be the original CNRG object. The operators in the tree of S are numbered according to the order in which they appear in the CNRG expression of S. Let I be the number of a "-" operator and J the number of a "*" operator in S. Let S' be the CNRG object obtained by replacing in S the I and J operators by "+". S is the sub-object of S' defined by using in the signature of S' a "1" in the Ith position, a "3" in the Jth position, and a "?" in all other positions. Indeed, "+₁" is equivalent to "-" and "+₃" is equivalent to "*". In the example of Figure 7, where $S = (A - B) - C$, we can define S' to be $(A + B) + C$, and obtain $S = S'_{11}$.

Using filters that combine other interpretations for operators I and J yields sub-objects of S' that are outside of S. Regions of these sub-objects may be constructed through filters and unions, as described in Subsection 3.10. Their topological relations with other regions may often be expressed

in terms of the existence and dimensionality of regions of CNRG objects obtained by applying topological and Boolean CNRG operators to these sub-objects.

For example, the existence and maximum dimension of selected regions in these sub-objects indicates contact or interference between features that have been subtracted by intersection or difference. In the above example, the dimensionality of S'_{33} indicates whether the portions of the two features, B and C, that lie inside A overlap (see Figure 7a), share a common face (see Figure 7b), or are disjoint (see Figure 7c).

4.0 ALGORITHMS FOR CNRG TREES

The most fundamental tool for interrogating a CNRG tree is an algorithm that classifies a point with respect to the CNRG object represented by the tree.

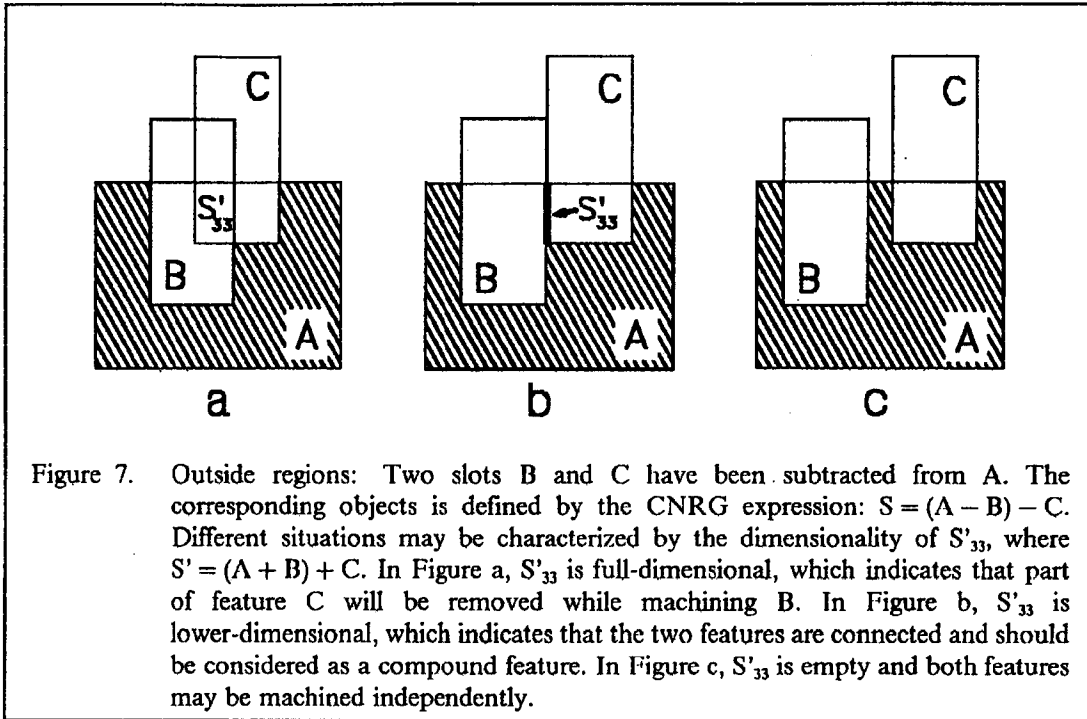
On a standard CSG tree, this operation is called PMC (for Point Membership Classification) [28] and returns a scalar that can take three values: IN, ON, or OUT, depending on whether the point is in the interior, the boundary, or the complement of the represented CSG solid. For points that lie on the boundary of several primitives, the result of PMC (with respect to sub-solids) is not only one of these three values, but also the *neighborhood* of the classified point with respect to the pointset represented by the corresponding node of the CSG tree [28,29]. The neighborhood is *full* if the point is in the topological interior of the solid or *empty* if the point is outside the solid. For points on the boundary of the solid, the neighborhood representation is more complex. The neighborhood of points that lie in the (2-D) interior of a solid's face may be represented by the surface that contains the face and by a single bit of information specifying whether the solid is on one side of the surface or on the other. For points that lie on an edge of the solid, but not on a vertex, an arrangement of faces around the edge, plus relative orientation information defines the neighborhood. Neighborhoods must be combined according to the CSG expression.

Since a CNRG object is typically composed of several regions, the notion of PMC must be extended to distinguish these regions. As explained earlier, any individual region of a CNRG object may be selected by attaching semantic interpretations to some of the operators of a CNRG tree. PMC (and other functions) may be computed independently for each region, or for all regions in parallel. As in CSG, PMC may be done by classifying the point with respect to the primitives and merging the results of these classifications upward in the CNRG tree. We do not discuss here point/primitive classification, and simply assume that it is available for all the primitives of the domain. (This is clearly the case for primitives defined as algebraic inequalities.) Assuming that PMC results (with respect to a particular region) are known for the operands (children) of a CNRG node, we define below how these results should be combined for each interpretation of the CNRG operators. We limit the discussion to two cases: *interior points*, when points are in general position (not lying on any curve, face, ... hyperface), and *hyperface points*, when points lie on a single hyperface. For points on more than one hyperface, one needs n-dimensional neighborhood representations and techniques for performing Boolean operations on such neighborhoods.

Neighborhood representation for CNRG are more complex than for CSG because CNRG objects may be non-regular sets. Observe that a point on the boundary of a regular set must be adjacent to a portion of the object's interior, but this need not be true for CNRG objects. The neighborhood representation must be able to distinguish between points that are ON the boundary *and* belong to the object and other points that are ON the boundary but do not belong to the pointset.

4.1 Interior points

Consider a point that does not lie on any boundary (with respect to \mathbb{R}^n) of any primitive. Such a point must be either in the topological interior of a region or in the interior of the object's complement. Therefore, the point's classification has only two possible values that distinguish whether the point belongs or not to the region. We represent these values, respectively, by Boolean *true* and *false* values. PMC for such points may be computed as follows. Let K_A (respectively K_B) be the



classification of a point P with respect to a particular region of A (respectively B). These regions have been selected by specifying, when necessary, the appropriate semantic interpretations for operators in the sub-trees of A and B. Also let K_{pA} (respectively K_{pB}) denote the classification of P with respect to the pointset, i.e. union of all the regions, of A (respectively B). Let K_C be the classification of the point with respect to a region of the parent node that corresponds to an operation on A, or on A and B. K_C can be expressed as a Boolean formula in K_A, K_{pA}, K_B, K_{pB} . We treat each interpretation of each operator as a separate case:

- Case \bar{A} : $K_C = \text{not } K_{pA}$
- Case sA : $K_C = K_{pA}$
- Case iA : $K_C = K_A$
- Case k_1A : $K_C = K_A$
- Case k_2A : $K_C = \text{false}$
- Case ∂_1A : $K_C = \text{false}$
- Case ∂_2A : $K_C = \text{false}$
- Case $A * B$: $K_C = K_A \text{ and } K_B$
- Case $A - B$: $K_C = K_A \text{ and not } K_{pB}$
- Case $A +_1 B$: $K_C = K_A \text{ and not } K_{pB}$
- Case $A +_2 B$: $K_C = K_B \text{ and not } K_{pA}$
- Case $A +_3 B$: $K_C = K_A \text{ and } K_B$

These formulae follow directly from the definitions of the operators. Note that classification of an interior point with respect to a region defined by k_2, ∂_1 , or ∂_2 is always false.

4.2 Hypersurface points

Let us now consider the case of a hypersurface point, P, i.e., a point that lies on a single $(n-1)$ -dimensional hypersurface of a primitive or possibly on the $(n-1)$ -dimensional intersection of several overlapping primitive hypersurfaces. P does not lie on any (transversal) intersection of two primitive hypersurfaces that are not subsets of the same hypersurface. We also assume that P does not lie on any cusp or singularity of the hypersurface. Let H denote the hypersurface that contains the primitive hypersurface(s) containing P.

The open n -dimensional neighborhood ball, N , of a hyperface-point, P , is divided into three sets by the hypersurface H : N_{inH} , the $(n-1)$ -dimensional intersection of N with H , and the two open disjoint sets, N_L and N_R of $N \setminus H$, one on each side of H . Because P is a hyperface point, the radius of the ball can be chosen such that these three sets are disjoint from any other hypersurface that bounds the primitives of the CNRG tree. Consequently, given any region of the CNRG object, each of these three sets is either entirely inside or entirely outside of it. Therefore, given P and H , the neighborhood of P with respect to any CNRG region may be represented by three "flags" (bits of information), each indicating whether the corresponding parts of B (N_{inH} , N_L , and N_R) lie inside or outside that region.

Assuming that H is consistently oriented, we refer to the two sides of H as *left* and *right*. The Boolean symbols K_{LC} and K_{RC} define whether there is material (of that particular region) around the classified point. K_{LC} refers to the left side of the hypersurface and K_{RC} to the right side. For example, $K_{LC} = \text{true}$ indicates that the region of C under consideration includes a hypervolume on the left side of H , in the neighborhood of P .

The symbols K_C , K_A , K_B , K_{pA} , and K_{pB} are defined as above. Furthermore, K_{LA} , K_{RA} , K_{LB} , and K_{RB} specify whether there is material of the selected regions of A and B around P on the corresponding side of H . The example of Figure 8 illustrates the meaning of these symbols.

K_{LpA} , K_{RpA} , K_{LpB} , and K_{RpB} have analogous meanings, but with respect to the entire pointsets of A and B .

Case sA: $K_C = K_{pA}$, $K_{LC} = K_{LpA}$, $K_{RC} = K_{RpA}$.

Case \bar{A} : $K_C = \text{not } K_{pA}$, $K_{LC} = \text{not } K_{LpA}$, $K_{RC} = \text{not } K_{RpA}$.

Case $A - B$: $K_C = K_A$ and not K_{pB} , $K_{LC} = K_{LA}$ and not K_{LpB} , $K_{RC} = K_{RA}$ and not K_{RpB} .

Case iA: $K_C = K_A$ and K_{LpA} and K_{RpA} , $K_{LC} = K_{LpA}$, $K_{RC} = K_{RpA}$.

Case $A * B$: $K_C = K_A$ and K_B , $K_{LC} = K_{LA}$ and K_{LB} , $K_{RC} = K_{RA}$ and K_{RB} .

Case $A +_1 B$: $K_C = K_A$ and not K_{pB} , $K_{LC} = K_{LA}$ and not K_{LpB} , $K_{RC} = K_{RA}$ and not K_{RpB} .

Case $A +_2 B$: $K_C = K_B$ and not K_{pA} , $K_{LC} = K_{LB}$ and not K_{LpA} , $K_{RC} = K_{RB}$ and not K_{RpA} .

Case $A +_3 B$: $K_C = K_A$ and K_B , $K_{LC} = K_{LA}$ and K_{LB} , $K_{RC} = K_{RA}$ and K_{RB} .

Case $\partial_1 A$: $K_C = K_A$ and not (K_{LpA} and K_{RpA}), $K_{LC} = K_{LpA}$, $K_{RC} = K_{RpA}$.

Case $\partial_2 A$: $K_C = \text{not } K_A$ and (K_{LpA} or K_{RpA}), $K_{LC} = K_{LpA}$, $K_{RC} = K_{RpA}$.

Case $k_1 A$: $K_C = K_A$, $K_{LC} = K_{LA}$, $K_{RC} = K_{RA}$.

Case $k_2 A$: $K_C = \text{not } K_A$ and (K_{LA} or K_{RA}), $K_{LC} = K_{LpA}$, $K_{RC} = K_{RpA}$.

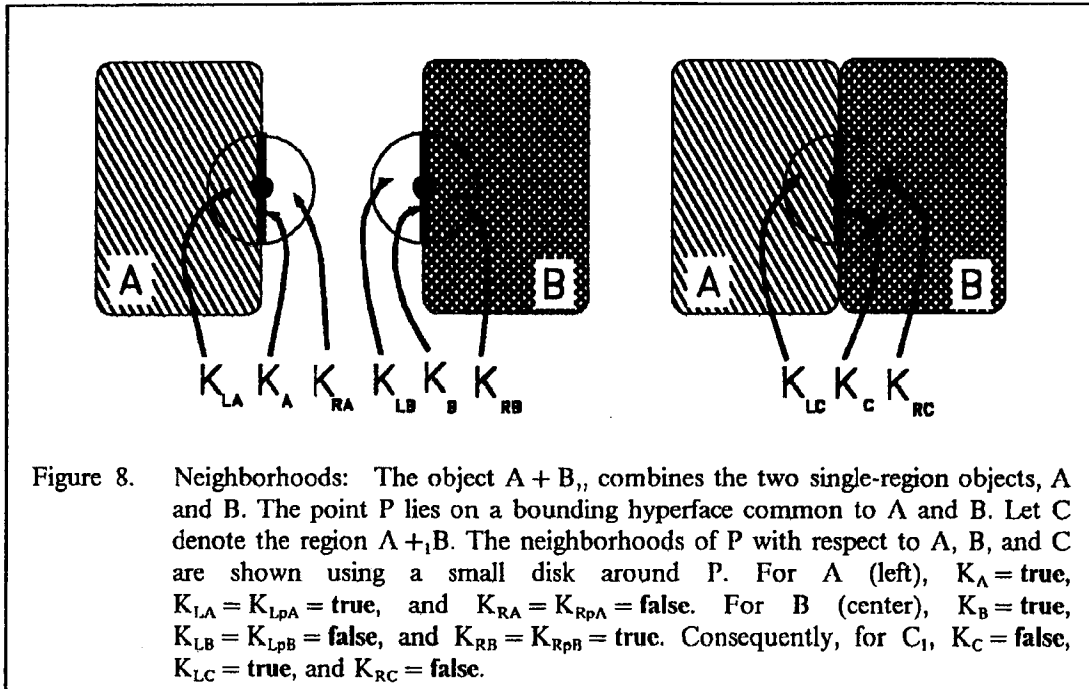
5.0 CONCLUSION

Dimensionally inhomogeneous pointsets of \mathbb{R}^n having missing boundary elements and with internal structures may be defined using a set of new Boolean and topological operators that construct and maintain a structural decomposition of pointsets into aggregates of disjoint regions which are non-regularized pointsets. These operators do not have all of the properties of their standard counterparts (for example, the union does not distribute over intersection). But they treat the pointsets that underlie the structured objects in a manner consistent with the corresponding standard operations. For example, $p(A + B) = pA \cup pB$ and $p((A + B) * C) = p((A * C) + (B * C))$.

The specification of inhomogeneous geometric objects may be stored in a CNRG (Constructive Non-Regularized Geometry) tree. Geometric algorithms originally developed for processing CSG trees in three-dimensions can be adapted for CNRG trees.

Since CNRG objects are aggregates of mutually disjoint regions, algorithms for processing CNRG trees must be able to distinguish among these regions. Any particular region of a CNRG object may be specified by its signature, i.e., a selection of the semantic interpretations for some operators of the CNRG expression. The signature indicates how the corresponding region can be expressed in terms of standard set-theoretic and topological operations applied to primitive pointsets.

Sub-objects, i.e. aggregates of regions, may be specified using a wild card in the object signature. These sub-objects are CNRG objects and thus can be used in Boolean and topological operations



and for region selections. Sub-objects may be obtained through a filter operation parameterized by the object's signature. Union of sub-objects suffice to express any combination of an object's regions.

A point can be classified with respect to a region of a CNRG object by a recursive descent, divide and conquer algorithm, which classifies the point with respect to the CNRG primitives (i.e., the leaves of the tree), and combines the results according to the CNRG expression of each region. Formulae for combining point classifications are presented in this paper for points that do not lie on the transversal intersection of two or more primitive-bounding hypersurfaces.

Explicit, boundary-like representations of CNRG objects are supported by Selective Geometric Complexes (SGC) structures and other "non-manifold" representation models.

Algorithms for evaluating SGC representations *incrementally* from a CNRG tree have been proposed [23] as compositions of operations that *subdivide* the cells (i.e. connected manifold components of the regions) of two objects, *select* and mark the appropriate components, and/or *simplify* the structure by deleting unmarked components and merging marked ones when appropriate. An implementation of these algorithms is currently under way.

Algorithms that *directly* evaluate boundary elements of the individual regions of 3-D CNRG objects and that perform Set Membership Classification with respect to regions are also under study. They involve more elaborate versions of the classification procedures discussed in this paper.

In summary, the CNRG methods discussed in this paper extend CSG representations and algorithms beyond solids to the domain of inhomogeneous objects which have "non-manifold topologies".

6.0 ACKNOWLEDGEMENTS

This work has benefited considerably from one of the authors' collaboration with Michael O'Connor on SGCs and with Paul Borrel on the applications of non-regularized structures to feature-based modelling.

The authors also wish to thank Paul Borrel, Josh Mittleman, Ari Rappaport and John Woodward at IBM; and Amitabh Agrawal at USC for their comments on the original manuscript.

7.0 REFERENCES

- [1] A.A.G. Requicha, "Representations for rigid solids: Theory, methods, and systems", *ACM Computing Surveys*, vol. 12, no. 4, pp. 437-464, December 1980.
- [2] I.C. Braid, "The synthesis of solids bound by many faces", *Communications of the ACM*, vol. 18, no. 4, pp. 209-216, April 1975.
- [3] C.M. Eastman and K.Preiss, "A review of solid shape modelling based on integrity verification," *CAD*, Vol. 16, No. 2, pp. 66-80, March 1984.
- [4] A.A.G. Requicha, "Mathematical models of rigid solid objects", Tech. Memo. No. 28, Production Automation Project, Univ. of Rochester, November 1977. (Available from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)
- [5] W. S. Massey, *Algebraic Topology: An Introduction*, New York: Harcourt, Brace and World, Inc., 1967.
- [6] R.C. Hillyard, "The BUILD group of solid modellers", *IEEE Computer Graphics and Applications*, vol. 2, no. 2, pp. 43-52, March 1982.
- [7] C.M. Brown, "PADL-2: A technical summary", *IEEE Computer Graphics and Applications*, vol. 2, no. 2, pp. 69-84, March 1982.
- [8] G.M. Koppelman and M.A. Wesley, "OYSTER: A study of integrated circuits as three-dimensional structures" *IBM Journal of Research and Development*, vol. 27, no. 2, pp. 149-163, March 1983.
- [9] S.A. Cameron, "Modelling solids in motion", PhD thesis, University of Edinburgh, (available from the Department of Artificial Intelligence), 1984.
- [10] R.N. Wolfe, M.A. Wesley, J.C. Kyle, F. Gracer, W.J. Fitzgerald, "Solid Modelling for Production Design" *IBM Journal of Research and Development*, vol. 31, no. 3, pp. 277-295, May 1987.
- [11] J.R. Rossignac, "Issues on feature-based editing and interrogation of solid models", IBM Research Report RC 15256, May 1990. *Computers and Graphics*, Vol. 14, No. 2, pp. 149-172, 1990.
- [12] K.J. Weiler, "Polygon Comparison using a Graph Representation", *Proc ACM SIGGRAPH'80*, pp. 10-19, July 1980.
- [13] G. Vanecsek and D. Nau, "Non-regular decomposition: An efficient approach for solving the polygon intersection problem", *Proc. Symposium on Integrated and Intelligent Manufacturing at the ASME Winter Annual Meeting*, pp. 271-279, 1987.
- [14] D.P. Dobkins and M.J.Laszlo, "Primitives for the Manipulation of three-dimensional subdivisions", Third ACM Symposium on Computational Geometry, Waterloo, Canada, pp.86-99 June 87.
- [15] M.J. Laszlo, "A data structure for manipulating three-dimensional subdivisions", PhD dissertation, Report CS-TR-125-87, Department of Computer Science, Princeton University, August 87.

- [16] P. Lienhardt, "Extension of the notion of MAP and subdivisions of a three-dimensional space", Cinquième Symposium sur les Aspects Théoriques de l'Informatique, STACS 88, Bordeaux, February 1988.
- [17] K.J. Weiler, "The radial edge structure: a topological representation for non-manifold geometric modeling", in *Geometric Modeling for CAD Applications*, M. Wozny, H. McLaughlin, and J. Encarnacao, Edts., Springer Verlag, pp. 37-68, May 1986.
- [18] E. Levent Gursoz and F.B. Prinz, "Node-based Representation of Non-Manifold Surface Boundaries in Geometric Modeling", To appear in "Geometric Modeling for Product Engineering, Proceedings of the 1988 IFIP/NSF Workshop on Geometric Modelling", Rensselaerville, NY, September 18-22, 1988. M. Wozny, J. Turner, and K. Preiss, eds., North-Holland, 1989.
- [19] P. Lienhardt, "Subdivision of n-dimensional spaces and n-dimensional generalized maps", 5th ACM Symposium on Computational Geometry, Saarbruchen, West Germany, June 1989.
- [20] H. Bieri and W. Nef, "Elementary set operations with d-dimensional polyhedra". Institut für Informatik und angewandte Mathematik, Universität Bern, Länggassstrasse 51, CH-3012 Berne. Lecture given at the 4th Workshop for Computational Geometry, University of Würzburg, March 24/25, 1988.
- [21] C. Cattani and A. Paoluzzi, "Solid Modeling in any dimension", Dip. di Matematica, Università "La Sapienza", P.le A. Moro 2, 00185 Roma. Unpublished manuscript submitted to TOG. 1989.
- [22] F. Arbab, "Set models and Boolean operations for solids and assemblies", Technical Report CS-88-52, Computer Science Department, University of Southern California, Los Angeles, California, 90089-0782, July 1985.
- [23] J.R. Rossignac and M.A. O'Connor, "SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries", IBM Research Report RC14340, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, New York. January 89. Also in "Geometric Modeling for Product Engineering, Proceedings of the 1988 IFIP/NSF Workshop on Geometric Modelling", Rensselaerville, NY, September 18-22, 1988. M. Wozny, J. Turner, and K. Preiss, eds., North-Holland, 1989.
- [24] A.A.G. Requicha and S.C. Chan, "Representation of geometric features, tolerances and attributes in solid modellers based on constructive geometry", *IEEE Journal of Robotics and Automation*, vol. 2, no. 3, pp.156-166, September 1986.
- [25] J.R. Rossignac, P. Borrel, and L.R. Nackman, "Interactive design with sequences of parameterized transformations", *Proceedings of the Second Eurographics Workshop on Intelligent CAD Systems: Implementation Issues*, April 11-15, 1988, Veldhoven, The Netherlands, pp. 97-127. Also available as IBM Research Report RC 13740, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, May 1988.
- [26] K.J. Weiler, "Topological structures for geometric modeling", PhD Dissertation, Rensselaer Polytechnic Institute, August 1986.
- [27] A.A.G. Requicha and R.B. Tilove, "Mathematical foundation of constructive solid geometry: General topology of closed regular sets", Tech. Memo. No. 27a, Production Automation Project, Univ. of Rochester, June 1978. (Available from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)
- [28] R.B. Tilove, "Set membership classification: A unified approach to geometric intersection problems", *IEEE Trans. on Computers*, vol. C-29, no. 10, pp. 874-883, October 1980.
- [29] A.A.G. Requicha and H.B. Voelcker, "Boolean Operations in Solid Modelling: Boundary Evaluation and Merging Algorithms", *Proceedings of the IEEE*, Vol. 73, No. 1, January 1985.

Computer-Aided Design

volume 23 number 1 january/february 1991

Editor Professor John Woodwark

47 Stockers Avenue, Winchester, Hants SO22 5LB, UK

SPECIAL ISSUE: BEYOND SOLID MODELLING

Guest Editor: Dr Jarek Rossignac

2 Editorial

4 Construction and optimization of CSG representations

V Shapiro and D L Vossler

21 Constructive Non-Regularized Geometry

J R Rossignac and A A G Requicha

33 Boolean set operations on non-manifold boundary representation objects

E L Gursoz, Y Choi and F B Prinz

40 Extrusion and boundary evaluation for multidimensional polyhedra

V Ferrucci and A Paoluzzi

51 Integration of polynomials over n -dimensional polyhedra

F Bernardini

59 Topological models for boundary representation: a comparison with n -dimensional generalized maps

P Lienhardt

83 Some techniques for visualizing surfaces in four-dimensional space

C M Hoffmann and J Zhou

92 A relational graphical editing method for PCB design

Shi Kaijian and Sun Jian

Automatic routing methods used in CAD electronics design systems cannot guarantee 100% connection in practice. An editor is essential

96 Calendar

Front cover picture: Detail of Boolean operations between sphere and honeycomb-like structure by Gursoz, Choi and Prinz (p 38)

Managing Editor Marija Vukovojac

Group Editor Karen Panaghiston

Applicant : Somashekar Ramachandran
Subrahmanyam
Serial No. : 10/651,452
Filed : August 29, 2003
Page : 34 of 35

Attorney's Docket No.: 15786-018001

Exhibit C

Lambda Research Corp., *Illuminations* newsletter, Vol. 2, Issue 1, Jan. 2000, pp. 1-4.

Lambda Research Corp.'s *Illuminations*

Volume 2 Issue 1

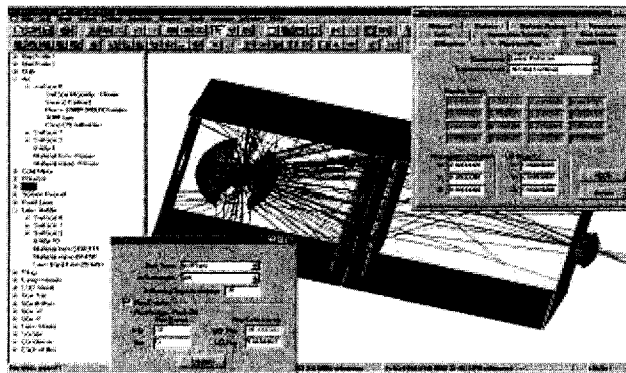
January 2000

Lambda Research's Latest Version of TracePro

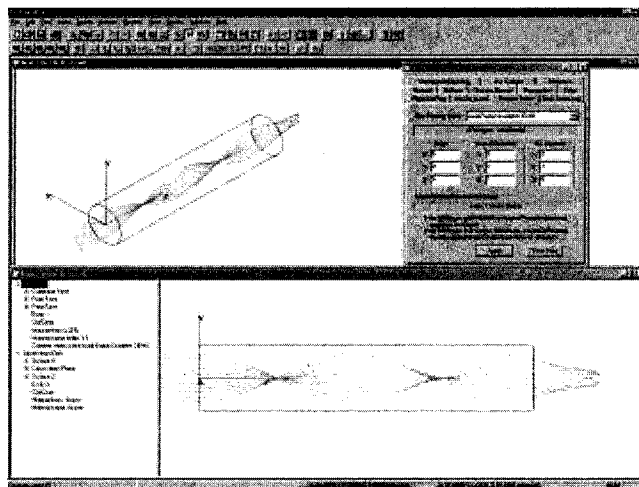
Lambda Research Corporation released Version 2.0 in September '99. This release raytraces, on the average, 10 times faster than its previous version. Further feature enhancements include a wizard to convert raw BSDF data into TracePro format (later article in this newsletter describes in more detail), over 3000 catalog lenses, a new bulb catalog, STL import capability, new expanded material catalogs and an optional healing husk to fix imperfect geometry.

In additional incremental updates since then, features have added specifically for the lighting design and biomedical markets. These features include bulk scattering for biomedical applications, updated luminaire design and analysis, a ray generation tool for bitmap images and a more powerful user interface.

The new features for the medical equipment market include ray tracing volume scatter in human tissue and general bulk scatter through any volume. TracePro 2.0.4 provides enhanced support of standard illumination output formats for the Illumination Engineering Society and Eulumdat, and DXF output for irradiance and illuminance maps. The new user interface displays additional data on-screen for multiple or single wavelengths at the user's discretion. And the optional bitmap module wizard translates any standard BMP, JPEG, GIF, or PNG file into a TracePro source file which can be ray-traced through any geometry and then displayed on any surface as an irradiance or illuminance output.



In the first quarter of 2000, TracePro Version 2.1 will be released featuring thin film stacks and ray tracing through gradient index materials.



What one of our customer's has to say about Version 2.0 — Fred Bushroe, President of Inov Inc., with over 14 years experience building optical hardware and developing products and systems, commented - *"TracePro 2.0 is incredibly fast compared to TracePro 1.4. I saw a speed increase of 26 times, raytracing a project containing reflective conic surfaces. Analysis that took an hour to trace now takes about two minutes. What's great about TracePro is the well thought out User Interface, and the fact that I routinely import and export complex solid models with ease. I can bring a radical 3D model or assembly in from SolidWorks, and focus my efforts on tracing rays and getting results, rather than spending days wrestling with model translation issues. TracePro's user interface is intuitive. This is important to me because I do many different things in my business. I can come back to the program after a month, and get right to work without having to relearn the user interface. Assigning optical properties to lenses and mechanical parts is a breeze."*

Our Mission Statement: *Establish a technology bridge between optical and mechanical engineers by providing state-of-the-art opto-mechanical design and analysis software, services and technical support.*

SolidWorks Partnership

Lambda Research has just signed on as a Research Partner with SolidWorks Corporation. Our objective is to improve the data transfer between Solidworks and TracePro.

How do I create a thin sheet using a macro in TracePro?

A thin sheet is a bounded area with zero thickness. In TracePro this may be build using the following ACIS Solid Topology: edges, wires, sheets and bodies. The boundary is build by a series of connecting linear edges which fully enclose the desired area. The loop of edges must not intersect.

In the Macro language we build up aa list of edges and convert them into a wire body. A wire body is a wire frame like solid without any surface data.

Once we construct the wire body which bounds our desired area we cover it with a surface. This becomes a single sided solid object. Single sided objects only have one surface normal (or can be thought to be infinite in thickness). To be useful to TracePro we can make the sheet double sided, with surface normals on either side of the now infinitely thin object.

Once constructed the thin sheet may be extruded and revolved to give it a non-zero volume.

The following example will construct a square boundary in the XY plane of size X wide and Y high.

We can assign some variables, X & Y in this case:

```
(define X 2)
(define Y 3)
(define edge-list
  (list
    (edge:linear (position 0 0 0) (position X 0 0))
    (edge:linear (position X 0 0) (position X Y 0))
    (edge:linear (position X Y 0) (position 0 Y 0))
    (edge:linear (position 0 Y 0) (position 0 0 0))))
```

The boundary is enclosed so we can create the wire body:

```
(define the-wire-body (wire-body edge-list))
```

Create a sheet body from the wire-body:

```
(define the-sheet-body (sheet:cover-wires the-wire-body))
```

Change the sheet body into a double-sided sheet body:

```
(sheet:2d the-sheet-body)
```

Done!

This can now be modifies within TracePro. To change the boundary you simply add more edges to the list.

The macro with minimal comments follows:

```
-----
(define X 2)
(define Y 3)
;
; Construct a list of the edges
;
(define edge-list
  (list
    (edge:linear (position 0 0 0) (position X 0 0))
    (edge:linear (position X 0 0) (position X Y 0))
    (edge:linear (position X Y 0) (position 0 Y 0))
    (edge:linear (position 0 Y 0) (position 0 0 0))))

; The boundary is enclosed so we can create the wire
body
;
(define the-wire-body (wire-body edge-list))
;
; Create a sheet body from the wire-body.
;
(define the-sheet-body (sheet:cover-wires the-wire-body))
;
; Change the sheet body into a double-sided sheet body
;
(sheet:2d the-sheet-body)
```

You can find more helpful hints on our web site in the TracePro App Notes Section.

TracePro's BSDF Wizard

The BSDF Wizard is a small 32-bit Windows utility, which can read ASTM files (as generated by a Schmitt Measurement Systems Scatterometer) and plot the graph of BSDF against β . β is calculated from θ using the Harvey-Shack Linear, Shift-Invariant Representation. A synthetic curve can then be matched to the data graph, thus giving an approximation of A, B, and g.

An ASTM file can be opened either by dragging the file into the graph rectangle area, or by clicking the "Open" menu item in the "File" menu. If there is a signature file specified in the ASTM file, a dialog box will ask to confirm opening the signature file.

The signature file is a second measurement that was taken with the instrument in an equivalent configuration, with no surface to be measured. This allows the background from the instrument to be subtracted out of the measurement. If the ASTM file does not specify a signature file, or specifies a non-existent or invalid signature file, a different signature file can manually be loaded by choosing the “Load Signature” menu item in the “File” menu.

There are several view options available in the “Graph” menu to aid in matching the synthetic curve to the data.

The Logarithmic β and Logarithmic BSDF selections control how the two axes are displayed in the graph. A log-log plot is best for mirror-like surfaces, whereas a linear-linear plot is best for rough surfaces. For mixed surfaces, such as glossy paint, a semi-log plot is usually best.

There are two display states for the input data: connected and non-connected. In the connected state, the data is represented as a continuous line. In the non-connected state, each data point is represented by a diamond shaped marker. The unconnected state helps in determining the distribution of the data, the connected state is useful for observing how closely the synthetic curve matches the data curve. The display state can be toggled by selecting “Connected” from the “Graph” menu.

The “Subtract Signature” selection in the “Graph” menu can be used to include or ignore the differences the signature makes. The show signature button can show or hide the signature graph.

A legend in the lower right hand corner of the BSDF Wizard dialog box, which indicates the colors of the data, signature, and synthetic curves.

At the upper and lower right hand side of the graph, there are two pairs of up- and down- arrow buttons that adjust the range of BSDF that is displayed in the graph window. Each pair of buttons increases or decreases the upper or lower limits of the graph. The synthetic curve can be adjusted to match the data points by dragging the two black control points until the sample curve appears to fit the data. If the control points are not within the viewable portion of the graph, the “Normalize” button will adjust the sample curve and graph limits so that all data points and control points are visible. For fine-

tuning, or to compare curves, the roll-off point and “A” can be specified manually by entering numbers in the edit boxes and clicking the “Apply” button to apply the changes to the synthetic curve. As the curve is adjusted, the “Integrated Scatter” (lower right hand side of dialog box) is calculated to show the total scatter of the synthetic curve.

The preferences can be adjusted by clicking on the “Preferences” menu item in the “File” menu. At this time, the only option is whether to confirm loading the signature file if detected. If this option is disabled, the signature file will automatically be loaded if it has a valid format. The “Properties” menu item in the “File” provides a view of additional information from the header in the data file and signature file.

To create an ASTM file manually: The first line of the file may contain anything at the beginning of the line. ASTM files generated by the SMS Scatterometer contain the file name, and the date and time the file was created. If there is a signature file associated with this ASTM file, then the first line should end with the words “signature file” (in all lowercase) followed by a colon and space, then the name of the signature file. The lines following the first line may contain comments about the contents of the ASTM file. Next, it is necessary to create the header block, which contains information about the instrument settings at the time of the measurement. The only header that is required by the BSDF Wizard is the “Incident Angle” field. The incident angle line should begin with the exact words “Incident Angle (deg)” followed by a colon, space, and the incident angle in degrees. The last item in the file, after the header block, is the BSDF measurement data. Immediately before the data should appear a line reading “Angle Data Follows [angle (deg), BSDF]” followed by a colon. The next line after this line should begin the data. Each line in the data section is one data point. Two numbers separated by spaces specify one data point. The first number is θ in degrees, and the second number is the BSDF. Scientific notation may be used.

Here is a minimal ASTM file:

```
signature file: SIGNATURE.CO2
This is the comment section, may omit
Incident angle (deg): 3
Angle Data Follows [angle (deg), BSDF]:
-92.0000    5.632E-06
-91.5000    1.428E-05
-91.0000    1.664E-05
. . .
```

Another Lambda Product-APEX™

APEX is an automatic interferogram analysis program for optical testing. It analyzes images from an optical interferometer to determine the aberrations present in the surface, element, or system under test. Pass/fail tolerances can be set in the program for go/no-go testing. Detailed reports can also be generated, along with graphical presentation of the aberrations, as well as Modulation Transfer Function (MTF) and Point Spread Function (PSF).

UPCOMING TRAINING

Illumination Analysis

Using TracePro

March 27-28, 2000

Two-Day Intensive Course

Stray Light Analysis

Using TracePro

March 29-31, 2000

Three-Day Intensive Course

Call for registration information (978)486-0766
*Watch for other course dates for Illumination Analysis
 and Stray Light Analysis*

Lambda Research represents the SOLEXIS family of products.

Recently Lambda Research has commenced reselling the database products of coatings and materials compiled by Stellar Optics Research International Corporation (SORIC) over seven years. This is the world's only data resource and selection tool for black, white, reflective and transmissive surfaces and materials for ground and space based applications.

Lambda Presented at SPIE Annual Meeting in Denver-July 1999

Our topics were:

- *Translation and Interchange for Mechanical CAD and Lens Design Data*
- *Polarization Models for Monte Carlo Ray Tracing*
- *End to End Electro-Optical Modeling Software*
- *Edge diffraction in Monte Carlo Ray Tracing*

If you would like to read the papers in their entirety, go to our Website and find them under TechNotes.

Distributors

Taiwan: InfoTek Information Systems Co., Inc.

No. 19, L-128

Chong-Shan N. Road. Soc. 2

Taipei, 104, Taiwan

Contact: Susan Lo (Susan@infotek.com.tw)

Phone: (81) 866 252 35377

Fax: (81) 886 251 12360

Japan: Future Instruments Trading Inc.

1-3 Nihonbashi-Ohdenmacho Chuo-Ku

Tokyo, 103, Japan

Contact: Tom Yamamoto

(tsutom@green.ocn.ne.jp)

Phone: (81) 3 366 67100

Fax: (81) 3 366 70094

Korea: Modern High Tech Co., Ltd.

Lifecomby Building, 61-4, Suite 418

Yeidodong, Yungdungpo-Ku, Seoul, So. Korea

Contact: H.B. Seo (modhitec@nuri.net)

Phone: (82) 278 60447

Fax: (82) 278 60814

Europe: LIGHT TEC

2 avenue Olbuis Riquier

Hyerres, 83400, France

Contact: Yan Cornil (sales@lighttec.fr)

Phone: (33) 4 94 01 37 47

Fax: (33) 4 94 01 37 46

**See Lambda Research at these
Upcoming Shows in 2000**

Jan. 25-27 - Photonics West, San Jose, CA

Booth # 1209

March 6-9 - SAE, Detroit, MI

March 19-23 - LightFair, Frankfurt, Germany

April 10-11 - OptoSouthwest, Albuquerque, NM

May 9-11 - Lightfair, New York, NY

May 16-18 - SID, Long Beach, CA

June 27-30 - Optatec, Frankfurt, Germany

August 1-3 - SPIE Annual Meeting, Denver, CO

September 10-15 - CLEO, Nice, France

November 6-8 - Photonics East, Boston, MA

Watch for Upcoming Announcements!

Applicant : Somashekar Ramachandran
Subrahmanyam
Serial No. : 10/651,452
Filed : August 29, 2003
Page : 35 of 35

Attorney's Docket No.: 15786-018001

Exhibit D

Schild et al., "Open Data Exchange with HP PE/SolidDesigner," *Hewlett-Packard Journal*,
October 1995, pp. 35-50.

Open Data Exchange with HP PE/SolidDesigner

Surface and solid data can be imported from HP PE/ME30 and exchanged with systems supporting the IGES, STEP, and ACIS formats. Imported data coexists with and can be manipulated like native data.

by Peter J. Schild, Wolfgang Klemm, Gerhard J. Walz, and Hermann J. Ruess

HP PE/SolidDesigner supports the coexistence of surface data with solid data and provides the ability to import and modify surface and solid design data from a variety of CAD systems. Backward compatibility with HP PE/ME30 preserves the investment of existing HP customers. Using improved IGES (Initial Graphics Exchange Standard) import capability, both surface and wireframe data can be imported. Surface data and solid data can also be imported and exported using the STEP (Standard for the Exchange of Product Model Data) format. Once imported, this data can coexist with HP PE/SolidDesigner solid data. It can be loaded, saved, positioned, attached to, managed as part and assembly structures, deleted, and used to create solids. Attributes such as color can be modified. If the set of surfaces is closed, HP PE/SolidDesigner will create a solid from those surfaces automatically.

HP PE/SolidDesigner 3.0 also allows solid parts and assemblies to be exported to ACIS-based systems using Version 1.5 of the ACIS® SAT file format. This feature provides a direct link to other ACIS-based applications.

From PE/ME30 to PE/SolidDesigner

HP PE/ME30 is a 3D computer-aided design (CAD) system based on the Romulus kernel. To preserve the investment of existing customers it was required that the transition from HP PE/ME30 to HP PE/SolidDesigner be as smooth as possible. Therefore, an HP PE/ME30 file import processor is a integral component of HP PE/SolidDesigner.

In HP PE/ME30, 3D objects are built from analytic surfaces like cylinders, cones, spheres, planes, and toruses. The intersections of these surfaces can be represented as explicit analytic curves such as straight lines, circles, and ellipses, or implicitly by describing the surfaces involved and providing an approximation of the intersecting arc. Parabolic and hyperbolic intersections are represented implicitly.

HP PE/ME30 Native File Organization

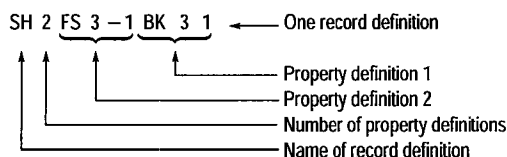
HP PE/ME30 supports the Romulus textual transmit format. The transmit file is not intended to be read by humans but the general structure can be examined. The file contains only printable characters, and real values are represented as

text strings. The full format of a transmit file consists of six different sections. These will be described using the example of a single cylinder positioned at the origin of HP PE/ME30's coordinate system with base circle radius 10 and height 20.

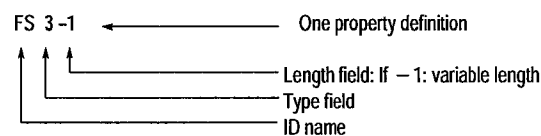
The first section, the header section, describes the environment, the machine type, the user login of the file creator, and the time and date when the model was created.

```
@* AOS
@* Machine type HP-UX
@* Transmitted by user_xyz on 27-May-94 at 13-06
```

The second section contains index and counting information related to the schema described in the third section. The schema defines the data structures used to represent the objects. It consists of a collection of record definitions. The following is an example of a record definition:



The following is an example of a property definition:



In the second section of the transmit file the number of record types, the numbers of record instances and property instances, the name of the schema, and its version and update number are supplied. The record instances and property instances contain the concrete data describing the model. The semantics and the sequence of data entities have to conform to the format specified by the corresponding record definition and property definition entities.

The information in the cylinder example file says that 11 instances of record definitions are supplied to describe the schema for the instance of the cylinder. For the actual object, 23 record instances built out of 115 property instances are used.

* A kernel is the heart of a modeling system. Currently, three kernels are used in various CAD systems. These are Romulus from Shape Data, Parasolid, an extension of Romulus, and the ACIS Kernel from Spatial Technology.

1
11
23 115
ROMDSCHMA 7 4
16

The third section, the schema section, contains the definition of the data structures used to represent the model. This section consists of the subset of record definitions from the HP PE/ME30 internal data structure schema that are needed to represent the model. The schema sections of files representing different models will be different. The schema section for the example cylinder is:

```
BY 19 UP 4 -1 SE 3 -1 TX 5 -36 FI 4 -6 CI 4 -12 PI 4 -8
GI 4 -1 SI 4 -3 TI 4 -3
RA 2 1 RN 2 1 ZI 4 -2 FN 1 1 CN 1 1 PN 1 1 TN 1 1
SN 1 1 ZN 1 1 NM 1 1
SH 2 FS 3 -1 BK 3 1
FA 8 UP 4 -1 AK 3 -1 RV 1 1 SF 3 1 SX 2 1 VR 3 -1
HA 2 -3 SL 3 1
VR 4 PT 3 1 BE 3 1 BV 3 1 FE 3 1
ED 2 CU 3 1 RV 1 1
CU 3 UP 4 -1 AK 3 -1 TR 3 1
TR 6 UP 4 -1 AK 3 -1 BK 3 1 EQ 2 -7 TS 3 2 TY 1 1
PT 4 UP 4 -1 AK 3 -1 CO 2 -3 GP 3 1
GP 5 UP 4 -1 AK 3 -1 BK 3 1 CO 2 -3 PX 3 1
SF 7 UP -1 SD 3 -3 AK 3 -1 BK 3 1 EQ 2 -7 SU 3 -5 TY 1 1
UA 3 OW 3 1 CL 1 1 II 1 *1
```

The fourth section contains, for each record type defined in the schema section, the number of data objects used for the transmission of the model. The sequence of numbers is identical to the sequence of record definitions used in the schema section. In the cylinder example, the object consists of one body built of one shell built of three faces. Four vertices, four coedges, two curve geometries, two edges, two points with two geometric point definitions, three surfaces, and one attribute are needed to represent the cylinder object. The file contents are:

1 1 3 4 4 2 2 2 2 3 1

The fifth section, the data section, contains the data structure instances. The contents of all records needed to represent the object are found in this section. To every record an integer record label is assigned. This number will be used in other record instances to point to the instance. In general the instances in the file appear in the order in which they are referenced by other entities. The data of an entity instance is not split. If forward references are contained in the instance definition the next instances can be found in exactly the same sequence as referenced. Because this rule applies recursively, newly referenced entities can be found first in the physical file sequence. If all references of an entity are resolved completely the next reference of the next higher level will be resolved. For the cylinder, the data section is:

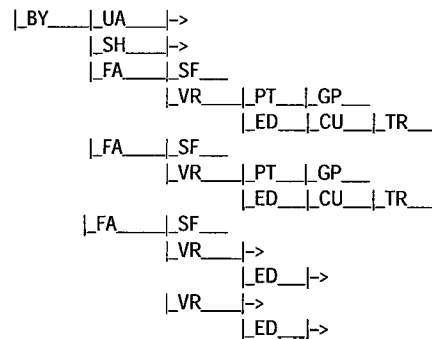
1
1 1 25 Color 1 2 0 3 3 F0 4 F1 5 F2 2 14 E0 15 E1 2 18 P0 19 P1 0 0 0 0.000001
0.00000000001 0 3 3 2 0 0 0 25 1 1 1 16777215 2 0 1 3 0 0 0 22 0 1 6 0 2 22
0 0 0 0 6 0 0 0 0 -1 0 1 6 18 10 6 10 18 0 0 0 20 20 0 0 0 3 0 10 0 18 10 14
0 14 0 0 16 16 0 0 0 7 0 0 0 0 -1 10 0 0 2 4 0 0 1 23 0 1 7 0 2 23 0 0 0 6
0 0 20 0 0 -1 0 1 7 19 11 7 11 19 0 0 0 21 21 0 0 0 3 0 10 20 19 11 15 1 15 0
0 17 17 0 0 0 7 0 0 20 0 0 -1 10 0 0 2 5 0 0 0 24 0 2 8 9 0 2 24 0 0 0 0 7 0 0
0 0 0 1 10 0 2 8 18 12 8 12 12 14 1 9 19 13 9 13 13 15 0

The sixth and last section contains, for each top-level object transmitted in the file, the corresponding root entity and its name. In the cylinder example only one object is transmitted. HP PE/ME30 supports user-named objects, but in this example an HP PE/ME30 default name, B0, has been used for the cylinder.

1
B0

Analyzing the Transmit File

Because the information content of an HP PE/ME30 file cannot be understood by simply looking at the file, several internal analysis tools are used to extract the information. Statistics showing the number of different curve and surface types give a first hint of the complexity of the file. A graphical presentation of the data instances of a file can be generated.



This reference structure can be read easily. The (cylinder) object in the file is a body (BY) which consists of a shell (SH) and three faces (FA). Shell and faces share the same hierarchy level. Each face consists of a reference to a surface (SF) and a start vertex (VR). Each start vertex is based on a geometric definition of a point (PT) and serves as the anchor vertex of an edge loop. A loop is not represented explicitly in the HP PE/ME30 exchange file. The implicit connection is done by a reference from a start vertex to the next and previous vertices in the loop. The edge (ED) entity represents the topological direction of the edge with respect to the loop. The curve (CU) entity is an intermediate instance on the way to the curve's geometry (TR).

If complete information from the data section is needed a translation tool is available that maps the data section to a format much more useful for human readers. The following extract describes how one of the faces and the corresponding surface component of the cylinder example are represented. The mapping from the data section to the readable format is also supplied.

For this component from the data section:

```
.....5 0 0 0 2 4 0 2 8 9 0 2 2 4 0 0 0 0 7 0 0  
0 0 0 1 1 0 0 2 .....
```

the corresponding translated part is:

5 = FA (Face owning (anchor) vertex), the properties are :

UP is	EMPTY	...	List of permanent universal attributes
AK is	EMPTY	...	Backpointer from element of feature
RV :	INTEGER = 0	..	Sense of face, edge geometry
SF :	POINTER = 24	..	Surface of face
SX :	REAL = 0	..	Hatching pitch
1 VR :	POINTER = 8	..	Anchor of face
2 VR :	POINTER = 9	..	Anchor of face
HA is	EMPTY	...	Hatch direction
SL :	POINTER = 2	...	Shell of face

24 = SF (Surface of face), the properties are :

UP is	EMPTY	..	List of permanent universal attributes
SD is	EMPTY	...	Surface supporting this surface definition
AK is	EMPTY	...	Backpointer from element of feature
BK :	POINTER = 0	...	Backpointer from assembly or body to token
1 EQ :	REAL = 0	...	Geometry definition
2 EQ :	REAL = 0	...	Geometry definition
3 EQ :	REAL = 0	...	Geometry definition
4 EQ :	REAL = 0	...	Geometry definition
5 EQ :	REAL = 0	...	Geometry definition
6 EQ :	REAL = 1	...	Geometry definition
7 EQ :	REAL = 10	...	Geometry definition
SU is	EMPTY	...	Surface supported by this surface
TY :	INTEGER = 2 (CYLINDER)	...	Geometry type

Import Module

The HP PE/ME30 to HP PE/SolidDesigner import interface is linked directly to the HP PE/SolidDesigner code. In HP PE/SolidDesigner's user interface it simply adds a button to the external filing menu. If a file name is specified, the processor is activated. Internally, several C++ classes are added to HP PE/SolidDesigner to represent the schema and instance entities of the HP PE/ME30 file. For every supported HP PE/ME30 record definition entity a class derived from a generic record instance object is defined. The most important member function of each of these classes is the convert function. This function performs the mapping of the HP PE/ME30 file object to the corresponding HP PE/SolidDesigner entity.

The three main components of the HP PE/ME30 to HP PE/SolidDesigner processor are a lookup table, a schema manager, and a set of classes to represent the supported HP PE/ME30 file entities.

The lookup table is part of the interface to an HP PE/ME30 file. The main task of this table is to manage the mapping of HP PE/ME30 file entities to already created corresponding HP PE/SolidDesigner entities. A lookup table is generated for every open HP PE/ME30 file.

A schema manager is initialized if a new HP PE/ME30 file is opened. It contains the schema section information found in the newly opened file. For every open file a corresponding schema manager is available to control the interpretation of the entities of the file.

The record instance class builds the third basic data structure of the processor. Record instances are generic containers to store all of the data objects that can be expressed by valid record definitions. The constructor of the record instance class calculates the entity type from the reference number and then allocates memory and reads in the properties from the file corresponding to the property definitions of the schema. For every supported HP PE/ME30 entity a separate C++ class is derived from the record instance class, but the generic constructor is used for all subtypes. The main differentiator between the classes is the convert function.

Conversion Process

The convert function of the record instance class itself is not called by the conversion process. Rather, every derived class implements its specific conversion function (in this sense the convert function is purely virtual in C++). The individual conversion function converts itself to an HP PE/SolidDesigner entity.

Conversion and the creation of new derived instances of the record definition class constitute a recursive process. If during an active conversion an unresolved (not already converted) reference is found the corresponding HP PE/ME30 file entities can be found as the next entities in the physical file (see the description of the data section). The conversion module then creates a new derived instance of the record instance class and forces the translation of this entity to a HP PE/SolidDesigner entity that can be used to complete the conversion of the current entity. The algorithm is as follows:

A reference to an HP PE/ME30 file entity is found:

Already "converted"? (lookup table search)

YES: Use the available conversion result

NO: Create the new derived class of record instance

Call the convert function

Attach the conversion result to the lookup table

Delete the instance to free the memory used

Use the newly generated conversion result to continue the conversion.

Nonanalytic Intersection Curves

The conversion for intersection curves is not done on the fly, but by a postprocessor after the rest of a body is converted completely. The convert routine for an intersection track simply collects the two intersecting surfaces and all available additional information found in the file to represent the intersection. The completion of the intersection curves is done by the convert function for HP PE/ME30 bodies. After a first intermediate topology of the new HP PE/SolidDesigner body is calculated and all analytic surfaces and analytic curves are attached to the created body, the calculation of the intersections begins.

The topology of the intersection between two surfaces in HP PE/SolidDesigner is not always the same as in HP PE/ME30 because different constraints on topology and geometry exist in the two modelers. For instance, it may be necessary to represent the single segment found in HP PE/ME30 as a sequence of different curves. In such cases the original topology has to be modified and some edges may be split. To find the appropriate intersection in HP

PE/SolidDesigner is mainly a selection process. In many cases two surfaces intersect at not only one but several distinct sections.

Consider the intersection of a cylinder with a torus in the case of perpendicular axes. Four possible intersection curves may be part of the model (see Fig. 1). In the HP PE/ME30 file additional help points are supplied to allow the correct selection. The direction of the intersection curve (the tangent to the curve) is not guaranteed to be the same in HP PE/SolidDesigner as in HP PE/ME30. Therefore the correct fit to the model is calculated and the resulting direction is reflected in the topology of the imported model.

Quality and Performance

To test the quality of the HP PE/ME30 import processor a large HP PE/ME30 test library has been compiled. It now contains more than 2300 examples of parts and assemblies. All of the test cases used during HP PE/ME30 development and support are included along with new user models consisting of recently acquired data from internal and external HP PE/ME30 users. An additional test matrix subtree was developed by creating base parts with critical features. In particular, all possible surface-to-surface intersections and various special cases have been generated.

The regression test procedure is to import HP PE/ME30 models from the test library part by part and perform the HP PE/SolidDesigner body checker operation on each. The loading time and the body checker result are collected in a reports file. A reports file can be analyzed by a shell script to supply a statistical summary of the current quality of the HP PE/ME30 interface. Because of the large amount of test data a complete test takes a long time. Therefore, an intermediate test is available. The complete test performs the basic load and check test on all currently available test models of the library directory. The intermediate test examines the reports file of the latest complete test and repeats all reported problems. It also repeats a random selection of the successful tests. At this time over 99% of the complete test conversions are classified as successful.

The performance of the import process for HP PE/ME30 files is mainly dependent on three variables: the size of the schema, the number of entities, and the number of intersections that have to be calculated:

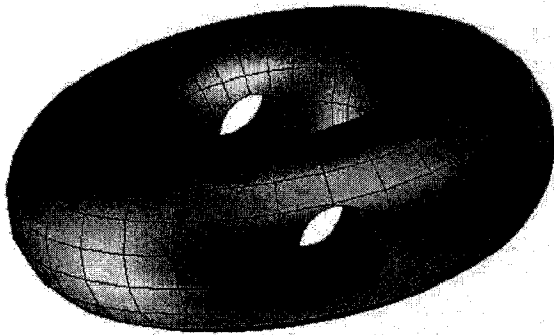


Fig. 1. Intersection of a torus and a cylinder.

$$\text{Load Time} \approx \text{Size} \times k_1 + \text{Entities} \times k_2 + \text{Intersections} \times k_3,$$

where $k_1 < k_2 < k_3$. The size of the schema section does not vary very much between different files and is normally relatively small compared to the size of the data section. The number of entities and the file size are strongly related. The calculation and selection of the nonanalytic intersection curves fitting the model is a relatively expensive component of the processor because a completely new representation of the data structure has to be generated.

Data Exchange Using IGES

An important task in computer-aided design is the transfer of the completed model to downstream applications and other CAD applications. These applications vary from finite element analysis and numerically controlled (NC) manufacturing to visualization and simulation. HP PE/SolidDesigner currently uses IGES 5.1 (Initial Graphics Exchange Specification) for file-based data exchange.

Because of the broad variety of receiving systems an IGES interface must be flexible so that the contents of the output file match the capabilities of the receiving system. It must be possible to transfer whole assemblies keeping the information on the parts tree, or only specific parts of a model, or even single curves or surfaces. This is achieved by a mixture of configuration and selection mechanisms.

An analysis of the IGES translators of many different systems showed that it is possible to classify them in four main categories:

- Wireframe Systems. These systems are only capable of importing curve geometry. This is typical for older CAD systems or 2D systems with limited 3D capabilities.
- Surface Systems Using Untrimmed Surfaces. These systems are capable of importing untrimmed surfaces and independent curve geometry. This is typical for low-end NC systems that need a lot of interaction to create tool paths and define areas.
- Surface Systems Using Parametrically Trimmed Surfaces. These systems are able to handle trimmed surfaces. Trimming is performed in the parametric domain of the surfaces. Periodic surfaces are often not handled or are incorrectly handled. Each surface is handled independently. This is typical for surface modelers and sophisticated NC systems.
- Topological Surface Systems and Solid Modelers. These systems are able to handle trimmed surfaces using 3D curves as trimming curves. They are able to handle periodic surfaces, nonplanar topology, and surface singularities. Connection between adjacent trimmed surfaces is maintained and the normal to the trimmed surface is important for inside/outside decisions. This is typical for advanced surface and solid modelers.

HP PE/SolidDesigner's IGES interface is designed to work in four output modes: wireframe, untrimmed, trimmed parametric, and trimmed. Each output mode represents one of the categories of receiving IGES translators. This has the advantage of giving as much information about the solid model as possible to high-end systems (trimmed, trimmed parametric), without burdening low-end interfaces with too much information. For some modes (trimmed parametric) more configuration parameters allow fine tuning to specific systems to

maximize the transfer rate. Each mode has a specific entity mapping that describes which IGES entities are used to describe the model (see Tables I, II, and III). Users can specify additional product related data and arbitrary comments for the start and global sections of the IGES file directly via the IGES output dialog box. Specific configurations can be saved and loaded so that the configuration has to be determined only once for each receiving system. Fig. 2 shows the IGES dialog menu.

To allow maximal flexibility in what is translated, the user is allowed to select assemblies, parts, faces, and edges and arbitrary combinations. All selected items are highlighted and the user can use dynamic viewing during the selection process. If the user selects assemblies, the part tree is represented with IGES entities 308 and 408 (subfigure definition and instance). Shared parts are represented by shared geometry in the IGES file.

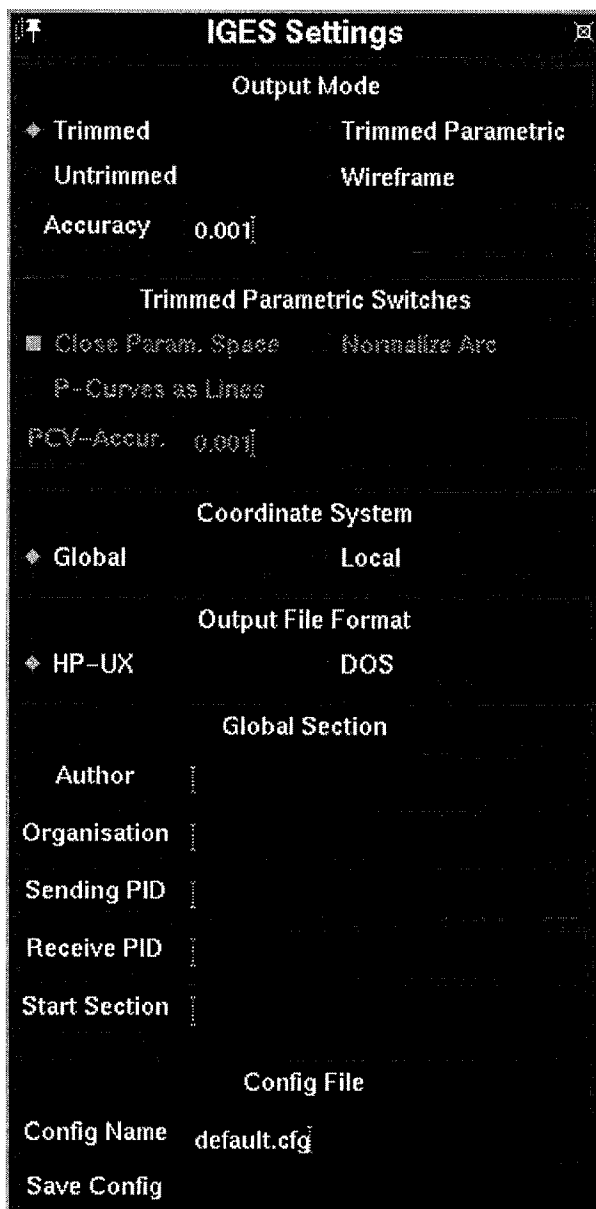


Fig. 2. HP PE/SolidDesigner IGES output dialog menu.

Table I
Curve Mapping

HP PE/SolidDesigner	IGES 3D Entity
Straight	Line (110)
Circle	Circular arc (100) with transformation
B-spline	Rational B-spline curve (126)
Intersection curve	Rational B-spline (126)
Parameter curve	Rational B-spline (126) or line (110)

Trimmed Mode

The trimmed mode is the closest description of the internal B-Rep (boundary representation) data structure of HP PE/SolidDesigner. It uses the IGES bounded surface entities 143 and 141 as the top element of the model description. Each selected face of the part maps to one bounded surface (entity 143) containing several boundaries (entity 141). Trimming of the surfaces is performed by 3D model space curves. To fulfill the requirements of the IGES specification of entities 141 and 143 some minor topological and geometrical changes of the HP PE/SolidDesigner internal model have to be made. Vertex loops are removed, propedges on toruses are removed, and intersection curves are replaced by B-spline approximations.

Because the IGES bounded surface entity 143 does not have any information about topological face normals, the surfaces are oriented so that all geometrical normals point to the outside of the part (Fig. 3). Thus, enough information is put into the IGES file that a receiving system can rebuild a solid model from a complete surface model.

Untrimmed Mode

The untrimmed mode contains basically the same information as the trimmed mode. For each face the untrimmed surface plus all trimming curves are translated. But instead of explicitly trimming the surfaces with the appropriate entities, surface and trimming curves are only logically grouped together. This usually requires manual trimming in the receiving system, and is only suited for some special applications.

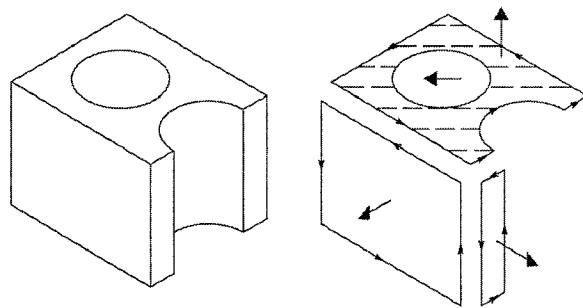


Fig. 3. (left) Solid model. (right) Surface model with normals.

Table II Surface Mapping		
HP PE/SolidDesigner	IGES 3D Entity (trimmed and untrimmed)	IGES 3D Entity (trimmed parametric)
Plane	Plane (108)	Ruled surface (118)
Cylinder	Surface of revolution (120)	Ruled surface (118)
Sphere	Surface of revolution (120)	Surface of revolution
Torus		
Cone		
Spun B-spline		
B-spline surface	B-spline surface (128)	B-spline surface (128)
Parallel swept B-spline	Ruled surface (118)	Ruled surface (118)

Trimmed Parametric Mode

The trimmed parametric mode uses the IGES trimmed parametric surface entity (144) and the curve on parametric surface entity (142) as representations of a trimmed surface. These entities have been established in the IGES standard for a longer time than entities 143 and 141 or the trimmed mode. For this reason they are more commonly used. The main difference from the trimmed mode is that the trimming is performed in the parametric domain of the surfaces. Each surface must have a parametric description that maps a point from the parameter domain D (a rectangular portion of 2D space) to 3D model space:

$$S(u,v) = (X(u,v), Y(u,v), Z(u,v)) \text{ for each } (u,v) \text{ in } D.$$

$$D = \{ \text{all } (u,v) \text{ with } u_{\min} \leq u \leq u_{\max}, v_{\min} \leq v \leq v_{\max} \}.$$

The following conditions apply to D:

- There is a continuous normal vector in D.
- There is a one-to-one mapping from D to 3D space.
- There are no singular points in D.

Furthermore, trimming curves in 2D space must form closed loops, and there must be exactly one outer boundary loop and optionally several inner boundary loops (holes). Fig. 4 illustrates parameter space trimming.

These restrictions make it clear that there will be two problem areas when converting HP PE/SolidDesigner parts to a parametric trimmed surface model: periodic surfaces and surface singularities.

On full periodic surfaces like cylinders, HP PE/SolidDesigner usually creates cylindrical topology. There will not necessarily be exactly one outer loop. Furthermore, 3D edges can run over the surface seam (the start of the period) without restriction. This leads to the situation that one edge may have more than one parametric curve (p-curve) associated with it. Also the p-curve loops may not be closed even if the respective 3D loop is closed. Fig. 5 illustrates this situation.

HP PE/SolidDesigner avoids this problem by splitting periodic surfaces along the seam and its antiseam. The seam and antiseam are the isoparametric curves along the parameters u_{\min} and $u_{\min} + u_{\text{period}}/2$. Thus, one face may result in

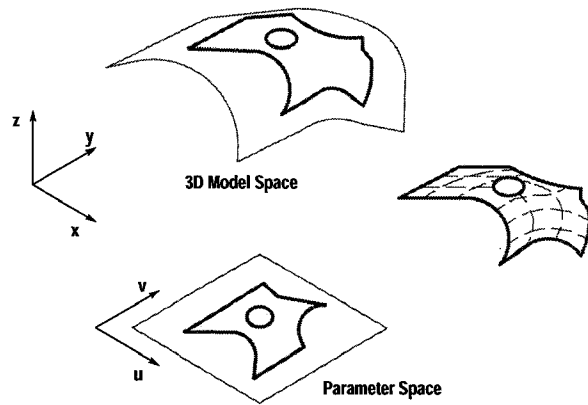


Fig. 4. Trimming in parameter space (p-space).

Table III Model Mapping				
Entity	Trimmed	Trimmed Parametric	Untrimmed	Wireframe
Parts and Assemblies	308+408	308+408	308+408	308+408
Faces	Entity 143	Entity 144	Entity 402	
Loops	Entity 141	Entity 142	Entity 102	
Edge+Base Curve	Curve Entity	Curve Entity	Curve Entity	Curve Entity
Base Surface	Surface Entity	Surface Entity	Surface Entity	None

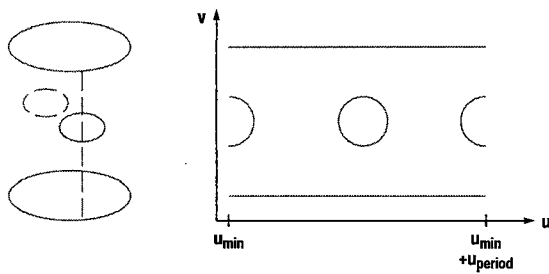


Fig. 5. Cylinder topology in 3D and p-space.

two or four parametrically trimmed surfaces (u- and v-parametric surfaces (toruses)) in the IGES model. Fig. 6 illustrates this situation.

Another problem with parametric trimmed surfaces are surface singularities. Singular points are points where the surface derivatives and normal are not well-defined. For such points there is not always a one-to-one mapping from 2D parameter space to 3D model space. This means there is an infinite set of (u,v) points in parameter space that result in the same 3D model space point. Such singularities are easily created by rotating profiles around an axis where the profile touches the axis. Examples are cones, spheres, degenerated toruses, triangular spline patches, and so on (see Fig. 7).

HP PE/SolidDesigner is designed to handle singularities as a valid component of a model. They are marked with a vertex if they are part of a regular loop or with a special vertex loop if they are isolated from the remaining loops. However, it is not possible to express singularities in trimmed parametric surfaces legally in IGES.

To resolve this issue we reduce the singularity problem to the problem of the valid representation of triangular surfaces. The splitting algorithm just described is applied so that all singularities are part of a regular loop. Thus, we are always faced with the situation illustrated in Fig. 8.

Each singularity of a face is touched by two edges, one entering and one leaving the singular vertex. Knowing how

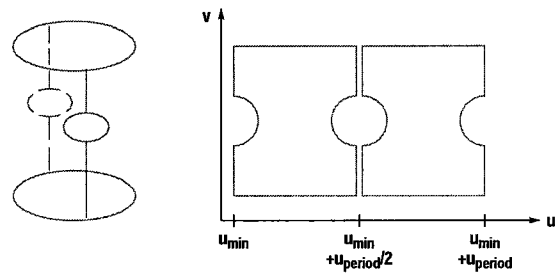


Fig. 6. Periodic surfaces in 3D and p-space after splitting.

triangular surfaces are handled in potential receiving systems, we offer four ways to export this kind of geometry. These are the four possible combinations of closed or open parameter loops and avoiding or using singularities.

Some systems do not need closed p-space loops, while others strictly expect them. If the closed option is chosen, the endings of the p-curves are simply connected with a straight line.

Geometrical algorithms usually become unstable near singularities. Some systems are not prepared to handle this situation and will fail. To avoid this, it is possible to shorten the parameter curves when entering or leaving a singular vertex and connect them at a numerically safe distance. This distance is measured in 3D space and is also configurable. It usually varies between 0.1 and 0.001. This will result in a surface where the region around the singularity is cut out. Fig. 9 illustrates the four possible singularity representations.

Wireframe Mode

For the wireframe mode HP PE/SolidDesigner also avoids the cylindrical topology, because in some cases information about shape would be lost (e.g., a full surface of revolution). After applying the face splitting algorithm all edges of the selected faces and parts are translated. No surface information is contained in the resulting IGES file.

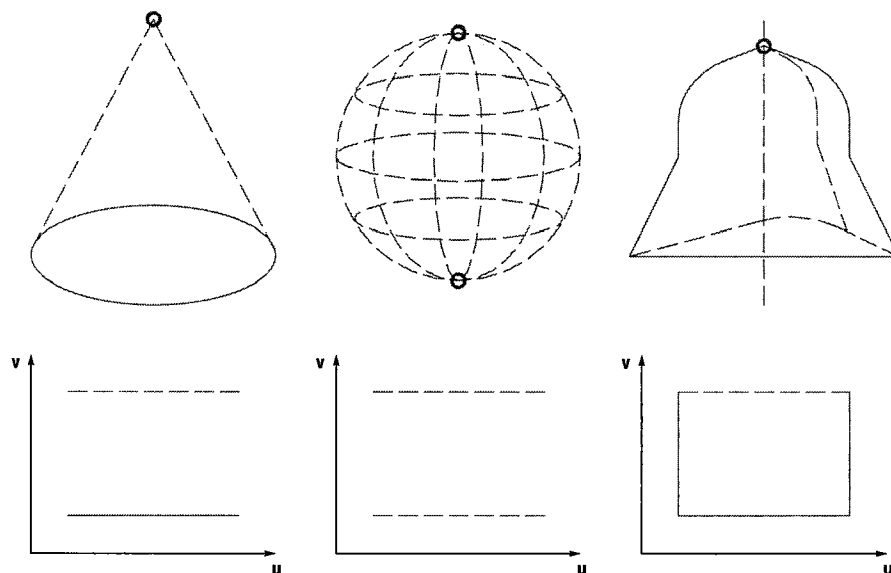


Fig. 7. Examples of surface singularities in parameter space.

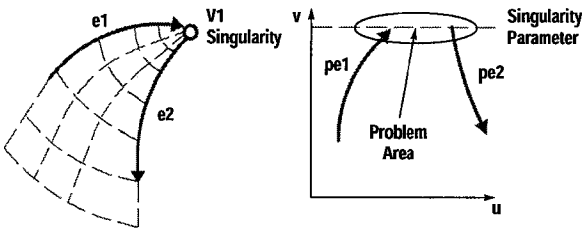


Fig. 8. Triangular surface situation.

Extracting Solid Information from Surface Models

IGES surface data from solid modelers often contains all surfaces of a closed volume or a connected face set. However, the connectivity between adjacent faces is lost. If the surface model fulfills some specific requirements it is possible for the receiving system to recompute this missing information. The following describes these requirements and shows how connectivity between faces can be reestablished. This method can be used to create a solid model from HP PE/SolidDesigner IGES output.

Automatic comparison of all boundary curves on coincidence or reverse coincidence would be a very time-consuming and numerically unstable task. However, it is common for the endpoints of the trimming curves of adjacent faces to be coincident within a very small accuracy. This makes it possible to identify trimming curves that share common start points and endpoints. If the two faces of these trimming curves have the same orientation one can try to connect the faces to a face set. For this task one must try to find a geometry for a common edge that fulfills the following accuracy constraints (see Fig. 10):

- The curve is close enough to surface 1.
- The curve is close enough to surface 2.
- The curve is close enough to curve 1.
- The curve is close enough to curve 2.

The first candidates for such a curve are the original trimming curves, curve 1 and curve 2. If either satisfies all four

requirements it is incorporated into both face descriptions and the connection is established. If neither curve can be used, one can try a combination of the two, or reduce the receiving system's accuracy.

This method fails if the face orientation is inconsistent or if adjacent faces do not share common start points and end-points.

Importing IGES Wireframe Data

IGES wireframe data can be easily imported into HP PE/SolidDesigner, since HP PE/SolidDesigner's kernel supports wire bodies. The modified wire data can be saved in HP PE/SolidDesigner's data format. Possible uses for this capability include migration from old-line systems to HP PE/SolidDesigner, interaction with different sources and suppliers, and communication with manufacturers.

In HP PE/SolidDesigner a wire is defined as a set of edges connected by common vertices. A body consisting only of wires is called a wire body. IGES 3D curve data is used to generate the edges of a wire body. This includes lines, circles, B-splines, polylines, and composite lines. IGES surface data such as trimming curves of trimmed surfaces are also used to generate edges. To simplify later solid model generation the axis and generatrix of a surface of revolution are also transformed into edges for the wire body. Since only edges have to be generated for a wire body, there are no accuracy problems as described above for IGES surface importation. On the other hand, information on B-spline surfaces is lost.

Wire data imported from an IGES file is collected into an assembly. The assembly gets the name of the IGES file. Any substructure of the IGES file like grouping in levels is transformed into parts within the assembly. Thus, hierarchical information contained in the IGES files is maintained within HP PE/SolidDesigner. The generated parts can be handled like any other part in HP PE/SolidDesigner. To distinguish

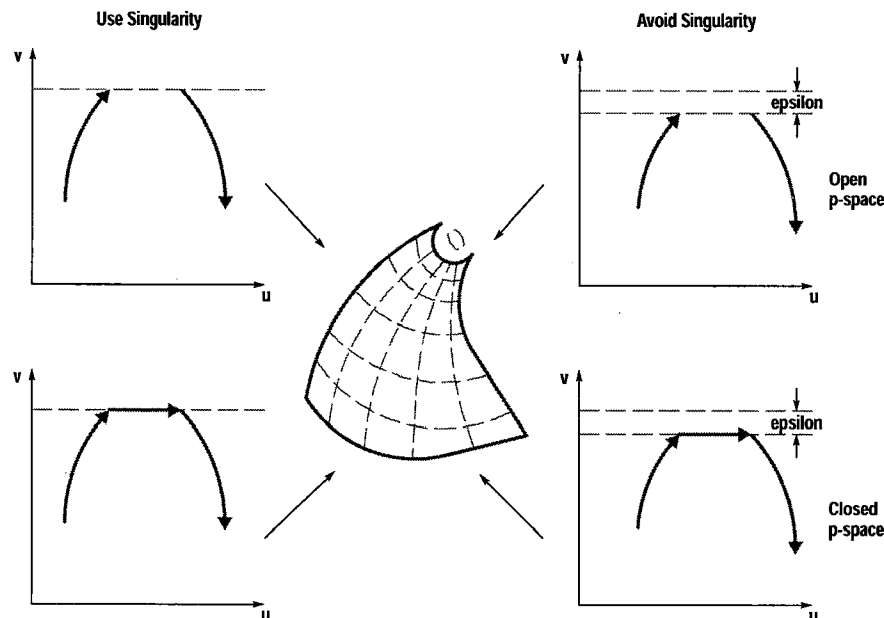


Fig. 9. Four possible singularity representations.

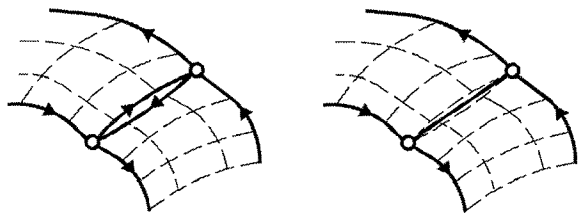


Fig. 10. Finding a common edge for adjacent faces.

wire parts, they can be colored. The options of HP PE/Solid-Designer's show menu work for the parts as well as the settings of the part container. A wire part can become the active part. The edges and vertices of a wire part are displayable, all browsers work with wire parts, and wire parts can be moved or become members of an assembly.

To build a solid model from a wire body, the edges and vertices of the wire body can be used to position a workplane. Then edges of the wire body can be selected and projected onto the workplane. The resulting profile can then be used to create a solid, for example by measuring an edge length needed for an extrude operation.

Fig. 11 shows an example of an IGES wireframe model with four parts and the resulting solid model. Automatic generation of solids from wires could be implemented but freeform surface information would probably be lost. The real benefit of wireframe import is for reference purposes.

STEP-Based Product Data Exchange

Manufacturing industries use a variety of national and industrial standards for product data exchange. These include IGES for drawing and surface exchange (international), VDA-FS for surface exchange (mainly the European automotive industry), and SET for drawing and surface exchange (France and the European Airbus industry). This variety of different incompatible standards causes a lot of rework and waste of valuable product development time which cannot be afforded if companies are to survive in the competitive marketplaces of tomorrow. Today's standards, originated in

the early 1980s, are no longer satisfactory for product data description and exchange. Standards like IGES or VDA-FS, which are limited to surface or engineering drawing exchange, do not adequately handle other explicit product data categories such as product structure or assemblies or geometric solid models.

Industry trends today are characterized by internationalization of manufacturing plants which are spread over the continents of the globe, and by lean production in which many parts are subcontracted or bought from local or international suppliers. National standards and incompatibilities between existing standards are obstacles to these trends and will have to be replaced by international standards.

Large companies in the aerospace and automotive industries in the U.S.A. and Europe have now taken the offensive towards the implementation and use of STEP (*Standard for the Exchange of Product Model Data*) as an international standard for product data exchange and access, starting in 1994. Companies such as BMW, Boeing, Bosch, General Motors, General Electric, Daimler-Benz, Pratt&Whitney, Rolls Royce, Siemens, and Volkswagen have been using STEP prototype implementations in pilot projects with promising results.

Ultimately, STEP is expected to meet the following requirements for an international product data exchange standard:

- Provides computer interpretable and standardized neutral product model data. Neutral implies compatibility with any CAD or CIM system that best fits the design or manufacturing task.
- Implements the master model concept for product data. The entire set of product data for a product with many single parts is kept in one logical master model which makes it possible to regenerate the product as a whole at a new manufacturing site. This means that product assemblies, including administrative data and bills of material, are handled.
- Provides completeness, conciseness, and consistency. This requires special data checking and validation mechanisms.
- Provides exchangeable product data without loss. The product data must be exchangeable from one CAD or CIM system to another without loss of data.

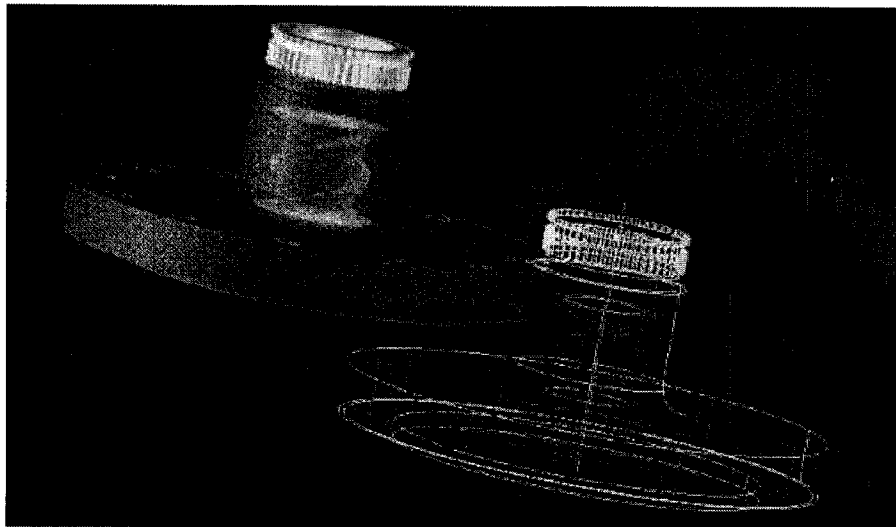


Fig. 11. Imported wire-body and the solid model constructed by HP PE/SolidDesigner.

- Provides long-term neutral data storage and interpretability. Product data is an important asset of a manufacturing company. The product data should be retrievable and interpretable by any CAD or CIM system after a long period of time, say 10 years or more. This is a significant challenge.

These requirements cannot be satisfied immediately. The STEP program also has shorter-term priorities for standardizing specific subsets of the product data. These include:

- The complete 3D geometric shape in the form of a 3D boundary representation solid model (B-Rep solids)
- Surface model and wireframe model data
- Product structure and configuration data.

Another priority is product documentation. An important goal is consistency of the engineering drawing with the 3D product geometry.

STEP Overview

STEP, the Standard for the Exchange of Product Model Data, is the ISO 10303 standard. It covers all product data categories that are relevant for the product life cycle in industrial use. STEP describes product data in a computer interpretable data description language called *Express*. The STEP standard is organized in logically distinct sections and is grouped into separate parts numbered 10303-xxx (see Fig. 12).

The resource parts of the standard describe the fundamental data and product categories and are grouped in the 1x, 2x, 3x, and 4x series. The Express data description language is defined in part 11. All other product description parts use the Express language to specify the product data characteristics in the form of entities and attributes. In addition to the product description parts there are implementation resources which are given in part 21, the STEP product data encoding scheme (the STEP file), and part 22, the Standard Data Access Interface (SDAI), which provides a procedural method for accessing the product data. There are different language bindings for part 22, such as C or C++ programming languages. The 3x series parts specify conformance requirements for STEP implementations.

Examples of STEP-standard resource parts are the fundamentals of product description and support (part 41), the geometrical shape (part 42), the product structure (part 44), material (part 45), the product presentation (part 46), tolerances (part 47), and form features (part 48). The application-specific resources are grouped in the 1xx series. Examples are drafting resources (part 101), electrical (part 103), finite element analysis (part 104), and kinematics (part 105). On top of the resource parts and application resources are the

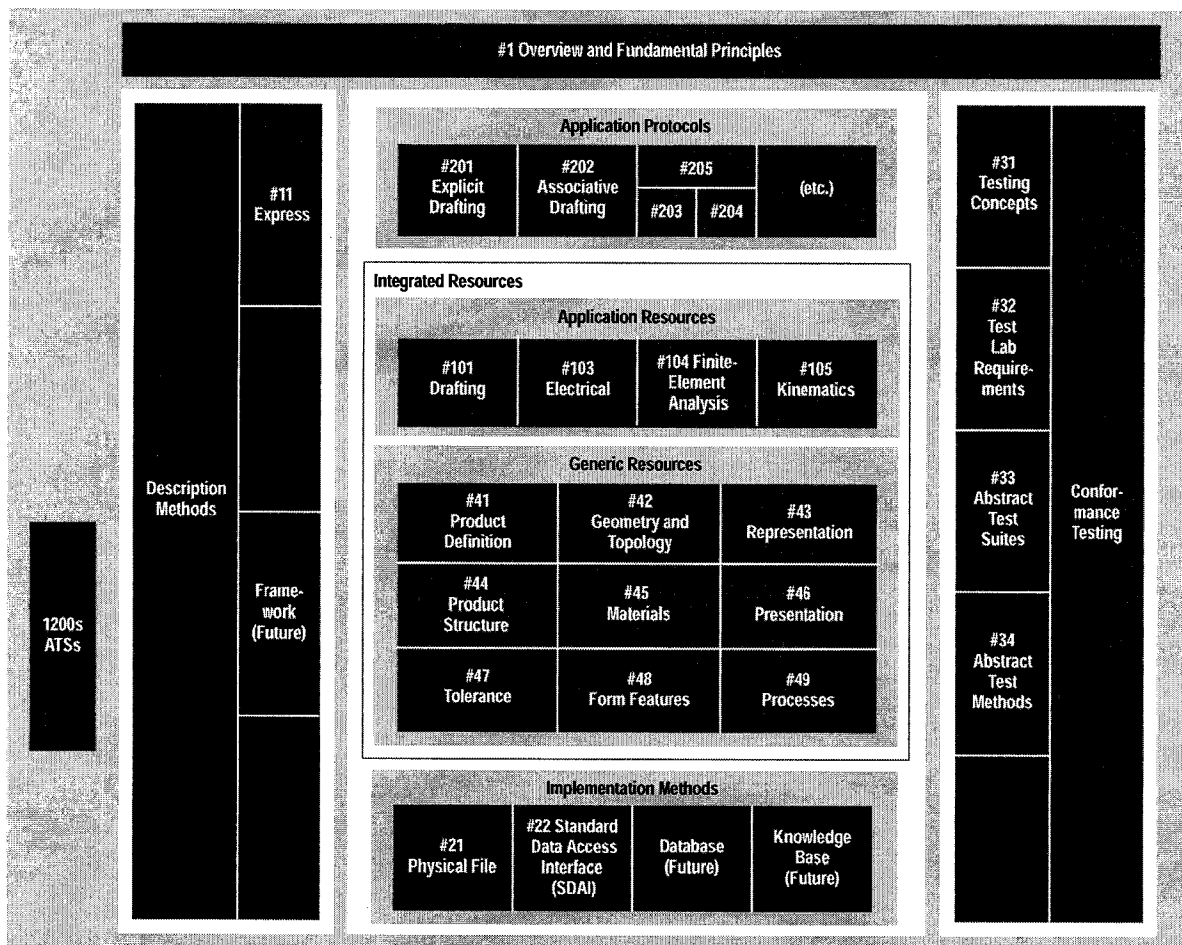


Fig. 12. Architecture of ISO 10303, Standard for the Exchange of Product Model Data (STEP).

application protocols (AP) which use the underlying resources in a specific application context, such as mechanical design for discrete part manufacturing, and interpret the resource entities in the application-specific context. STEP implementations for CAD or other computer-aided systems are based on application protocols. Application protocols are under definition for application areas like basic drafting, associative drafting, mechanical design, electrical design, shipbuilding, piping, architecture, and others. Here, we highlight just two examples, AP203 and AP214.

AP203: Configuration-Controlled 3D Design. AP203 was developed under the leadership of PDES Inc. It covers the major requirements for U.S.-based industries such as the aerospace industry for government and industrial manufacturing contracts. The product data covered in AP203 includes geometric shape (B-Rep solid models, surface models, wireframe models), product structure, and configuration management. AP203 is the underlying STEP specification for many CAD and CIM system implementations.

AP214: Core Data for Automotive Mechanical Design. AP214 has been developed by the automotive industry and covers product data categories relevant for the design and manufacturing of automotive parts and products. AP214, initiated in Germany and internationally supported, is still under finalization in parallel with its industrial implementation in CAD and CIM systems. The implementations have been coordinated and harmonized in the European ProSTEP consortium and the implementation is focused initially on the geometrical product descriptions (solid models, surface models) and product structure. However, all other kinds of product data categories relevant for mechanical design in the automotive industry (e.g., form features, materials, tolerances) are within the scope of AP214 and are going through the standardization process.

Initial Release

The initial release of STEP parts focuses on the most urgently needed kernel definitions of the standard, which cover the geometrical shape description, including all topological information, the product structure, and the configuration management data. Basic product documentation in the form of low-level engineering drawings is also covered. The parts included in the initial release are parts 1, 11, 21, 31, 41, 42, 43, 44, 46, 101, 201, and 203. The first two application protocols to become standards are AP201: Explicit Drafting and AP203: Configuration-Controlled 3D Design.

Upcoming releases of STEP will cover the next priorities in the area of drafting, such as AP202: Associative Drafting, materials, tolerances, form features, and parametrics, and other application protocols such as AP204: Mechanical Design Using B-Rep Solid Models and AP214: Core Data for Automotive Mechanical Design.

HP Involvement in STEP

HP has been working on the standardization of product model data since 1989 and has focused on the emerging international standard STEP for 3D product data. The product data focus has been on 3D kernel design data, completeness of topology and geometry, B-Rep solid models, and product structure and assemblies, as well as on associative drafting documentation. HP is an active member in organizations that

have an impact on the ISO STEP standard, and contributes to STEP through national standards organizations in the U.S.A. (e.g., NIST, ANSI) and Europe (e.g., DIN in Germany). Of particular interest are the organizations PDES Inc., PRODEX, and ProSTEP.

PDES Inc. HP has concentrated on three major areas of PDES Inc.'s STEP activities: mechanical design of 3D product data, associative drafting for CAD data, and electronic data definition and exchange.

The mechanical design initiative of the U.S. aerospace and aircraft industries, the automotive industry, and the computer industry resulted in STEP application protocol 203. HP, a PDES Inc. member in the U.S.A. and an ESPRIT CADEX member in Europe, contributed to the 3D geometric design definition of AP203 in a joint effort of PDES Inc. and CADEX. The AP203 3D geometries cover solid models, surface models, and wireframe models and are shared by other application protocols, thereby promoting interoperability between different application areas.

HP has also been actively supporting the U.S. initiative to define a good-quality standard for associative drafting documentation in STEP. Associative drafting, covered by AP202, is considered an integral portion of the product data for contractual, archival, and manufacturing reasons. For example, government contracts and ISO 9000 require that product data be thoroughly documented. This includes engineering drawing data of a product in addition to the 3D product data and the configuration data. Electronic design and printed circuit board design data are also covered in STEP.

PRODEX. In 1992 participants in the ESPRIT CADEX project demonstrated publicly the first B-Rep solid model transfer via STEP for mechanical parts in Europe. To develop this new technology the PRODEX project was founded in 1992 with the goal of developing STEP data exchange for CAD design, finite element analysis, and robot simulation systems. Twelve European companies joined the project. So far, the project's achievements include the definition of a STEP implementation architecture, the development of a STEP toolkit, and the development of STEP preprocessors and postprocessors.

Product data exchange between the different vendors is ongoing and shows very promising results for CAD-to-CAD data exchange, CAD-to-finite-element-system exchange, and CAD-to-robot-simulation-system exchange. The STEP standard has been further fostered by a joint effort with the ProSTEP project to develop AP214, in cooperation with the U.S., European, and Japanese automotive industries.

ProSTEP. ProSTEP is an automotive industry initiative for a highway-like STEP product model data exchange. In 1992 the German companies Bosch, BMW, Mercedes-Benz, Opel (GM), Volkswagen, and Siemens launched an initiative to bring the major CAD vendors together with the goal of implementing the first harmonized set of STEP product data exchange processors (product data translators) for industrial use in the automotive industry. The approach taken was to compile the user requirements, to build on the results and experiences of the ESPRIT CADEX project, and to launch at the ISO level a STEP application protocol, AP214, which covers the core data for automotive mechanical design.

The following CAD/CIM systems are involved in the project and have STEP data exchange processors either available or under development: Alias, AutoCAD, CADDs/CV-Core, CATIA, EUCLID3, HP PE/SolidDesigner, EMS-Power Pack, I-DEAS Master Series, SIGGRAPH STEPIntegrator, SYRKO, Tebis, ROBCAD, and others.

The initial focus in ProSTEP for STEP products is on design data exchange for 3D geometry: B-Rep solid models, surface models, and wireframe models. For migration from legacy systems, wireframe data needs to be supported, at least for data import. Communication with applications like numerical control (NC) programming systems today typically requires surface model data, although in the future more solid model data will be used. Initially, the HP emphasis is on bidirectional product model exchange (input and output) of 3D B-Rep and surface models.

STEP Tools Architecture

In STEP implementation projects, standardization has been extended beyond the product data to the STEP implementation tools. The CADEX, PDES Inc, PRODEX, and ProSTEP projects have all taken this approach.

A standardized STEP tool architecture provides the following benefits. These include shareability of tools between different implementors, shortened development time for STEP processor implementations (software development productivity gain), increased likelihood of compatibility between STEP implementations (differences in STEP definition interpretations are minimized), parallel development of tools (concurrent engineering), extendability of tools to track new standardization trends, increased flexibility (new STEP models require fewer code changes), and centralized maintenance of tools.

Fig. 13 shows the PRODEX STEP tools architecture. The functional blocks of a STEP toolkit or STEP development set are:

- STEP Standard Data Access Interface (SDAI),
- STEP Express compiler

- STEP file scanner/parser
- STEP file formatter
- STEP data checker
- STEP conversion tool.

The main interface to the STEP application is the STEP Standard Data Access Interface, which provides a computer programming language for dynamic access to the STEP data. Application-specific mapping and conversions are implemented on top of this interface.

The Express compiler conveys the product data descriptions contained in an Express schema (the metadata of the data model) to the toolkit. It contains an Express file reader and compiles the file contents to the internal representation of the data model. The SDAI is the recipient of the product data metamodel and uses the metamodel as a reference for the product instance data, which is imported through the STEP file scanner/parser.

The STEP file scanner/parser reads (scans and parses) the STEP instance data contained in a STEP data file and uses the currently valid metamodel for checking the syntax of the imported instance data.

The STEP file formatter formats the data to a part-21-conformant STEP file which is read from the SDAI by using the current valid metadata (e.g., a specific application protocol such as AP203).

The STEP data checker is a validation tool that checks the instance data currently in the SDAI based on the corresponding metadata model, which is also contained in the SDAI. The checking covers consistency checks like references between entities (e.g., existence dependency), and rule checking, which is covered in the metamodel. The checking is optionally applicable to the data in the SDAI. It is very helpful during the development of processors, for checking new metadata models, or for checking the first data imported from a new system.

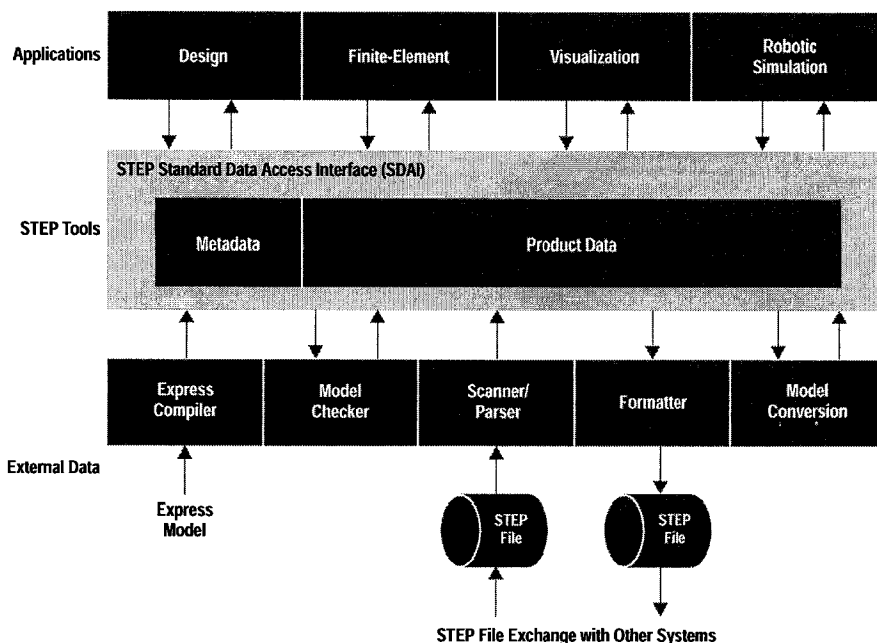


Fig. 13. PRODEX STEP tools architecture.

The STEP conversion tool is a pool of conversion functions (a library) that includes all kinds of geometrical, topological, and other model conversions. The focus is on geometrical conversions which are heavily used for data exchange between systems with different geometric modeling concepts. For example, one CAD system might use rational polynomial representations for its inherent geometric representation of curves and surfaces (e.g., NURBS, nonuniform rational B-splines), while the other might use nonrational representations (e.g., NUBS). In this case an approximation to the nonrational representation has to be applied, at the price of increasing the amount of data. For another example, a surface modeling system might export trimmed surface data with curve representations in 2D parameter space, whereas the receiving system might handle only 3D space curves. In this case the 2D parameter curves have to be evaluated and converted to 3D trimming curves in 3D space.

By using a STEP toolkit the requirements for the implementation of a STEP processor might be reduced to just the native data interface to the STEP tools, which consists of the data output to the SDAI (for the STEP preprocessor) and the data imported from the SDAI (for the STEP postprocessor).

The main task in linking a CAD system to the toolkit consists of defining and implementing the mapping between the system internal representation and the standardized entity representation in the schema of the standard (e.g., an application protocol).

HP PE/SolidDesigner STEP Implementation

The target application protocols for HP PE/SolidDesigner are initially AP203 and AP214, in which both solid and surface models are supported. In addition to the HP PE/SolidDesigner internal data models, the solid and surface models of other CAD systems are of major interest. With the introduction of STEP, B-Rep solid model data exchange comes into industrial use, representing a new technology shift. HP PE/SolidDesigner has its focus on solid models and is best suited for STEP-based bidirectional solid model exchange. However, surface models are also supported.

In addition to the geometric specifications, product information and configuration are covered in the implementation. In

this article, the geometric and topological mappings are discussed. The assembly, product structure, and administration mappings are not covered.

STEP Preprocessor (STEP Output)

The preprocessor exports the HP PE/SolidDesigner model data in a STEP file. The preprocessor takes care of the mapping of the HP PE/SolidDesigner model to the STEP model.

The internal geometrical and topological model of HP PE/SolidDesigner is in many respects similar to the STEP resources of part 42 of the STEP standard. Hence the mapping is often straightforward. On the other hand, there are data structure elements that are not mapped to the STEP model.

HP PE/SolidDesigner uses the following geometric 3D elements:

- Analytics: 3D surfaces such as planes, cones, cylinders, spheres, and toruses, and 3D curves such as lines, arcs, circles, and B-splines
- Nonanalytics: typically 3D elements such as B-spline curves and surfaces, and linear and rotational swept surfaces.

The topology used for the exchange of solid models is based on the manifold topology of STEP part 42. The elements used are manifold solid boundary representations, closed shells, faces, loops, edges, and vertices. The link between the topology and the geometry is given by references from faces to surfaces and from edges to curves. The geometrical points are referenced by vertices.

The HP PE/SolidDesigner STEP surface models are also based on topological representations. Special elements are used for surface models, such as shell-based surface models and closed and open shells. The other underlying topological elements are the same as in the solid models. The geometric representations of the surfaces are typically the same as in the solid model representations.

STEP Postprocessor (STEP Input)

The HP PE/SolidDesigner postprocessor supports the import of B-Rep solid models and surface models along with the necessary product structure data. The postprocessor is

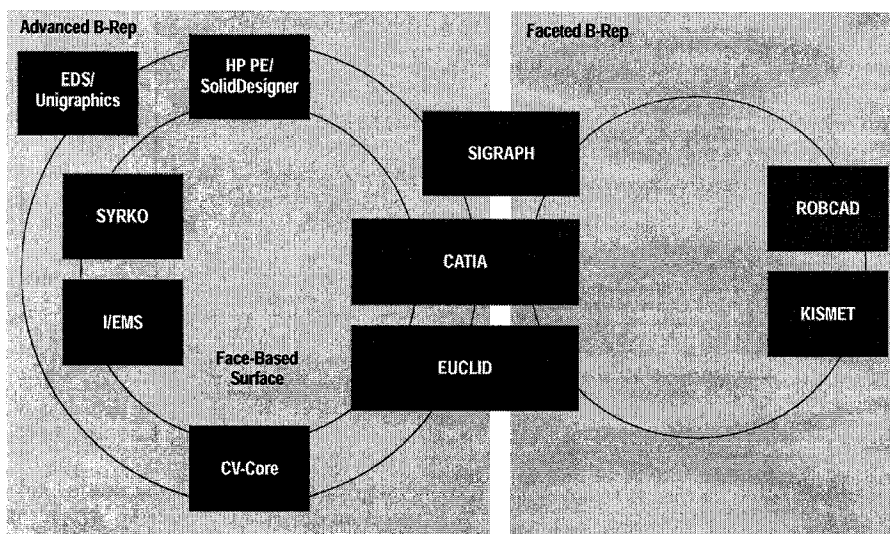


Fig. 14. Data exchange cycles between different CAD systems, including robot simulation systems, in the ProSTEP project.

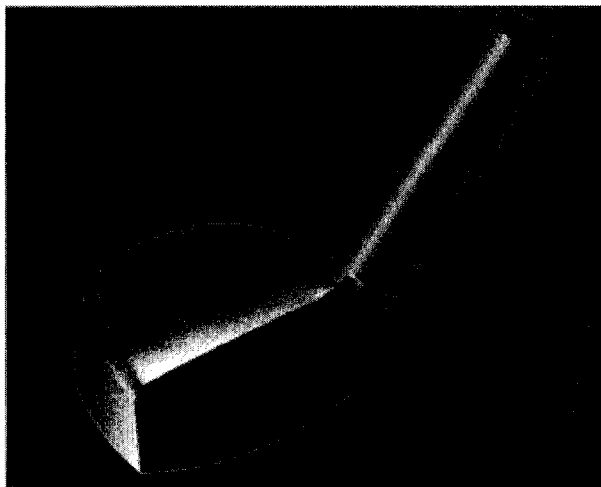


Fig. 15. Golf club solid B-Rep model imported into HP PE/SolidDesigner from CATIA (CAP-Debis).

capable of covering at least the functionality of the preprocessor so that it is possible to store and retrieve HP PE/SolidDesigner data in a STEP file representation (this is called the *short cycle test*).

The STEP postprocessor imports STEP files from other systems based on specifically supported application protocols. Postprocessing is one of the most difficult tasks in data exchange, especially when the data imported comes from a system that is very different from the receiving system. Potential problems arise in postprocessing if the sending and receiving systems have different accuracies, use different modeling techniques to generate the data, have different or missing surface connectivity, use different algorithms or criteria to determine surface intersections or connectivity, or use different model representations for similar model characteristics.

When surface models are imported, it cannot be guaranteed that they can be migrated to solid models even with user interaction. However, in special cases imported surface

models can be migrated to solid models without problems. In many cases imported surfaces provide boundary conditions for the solid model. In most cases the data can be used as reference geometry to check interference or provide dimensions for the solid models. For example, an imported surface set might represent the surrounding boundary geometry within which the final mechanical part has to fit without interference.

Importing surface models into HP PE/SolidDesigner is considered important and critical since many other CAD systems, especially legacy systems, often support only surfaces or wireframe models, not solid models. Therefore, the postprocessing of STEP surface models needs to cover a broader scope than the preprocessing. Sometimes, different surface representations are used in different application protocols, such as AP203 and AP214. Hence, different external representations may need to be mapped to one internal representation in HP PE/SolidDesigner.

In the initial implementation of the HP PE/SolidDesigner postprocessor, topology bounded surface models are supported. These provide the most sophisticated description of the connectivity of the individual surfaces used in a solid model. Geometrically bounded surface models are supported as a second priority.

The Accuracy Problem

When importing CAD data from other systems the accuracy of the data plays a key role and determines whether a coherent and consistent CAD model can be regenerated to represent the same kind of model in the receiving system.

Let's define the term accuracy. There are different accuracy or resolution values that must be considered in geometric modeling and CAD systems. For 3D space, a minimum linear distance value (a length resolution value) can be defined, which is the absolute distance between two geometric points that are considered to coincide in the CAD internal algorithms; this represents the zero distance. We'll call this value

* Often, CAD surface models are not consistent because the generating system lacks checking mechanisms or does not track connectivity. Very often, consistency and accuracy are the responsibilities of the user of the system rather than under system control.

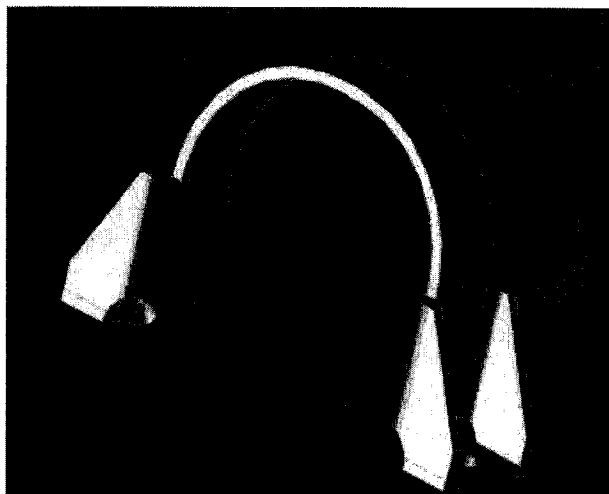


Fig. 16. Clamp solid B-Rep model imported from Unigraphics II (EDS).

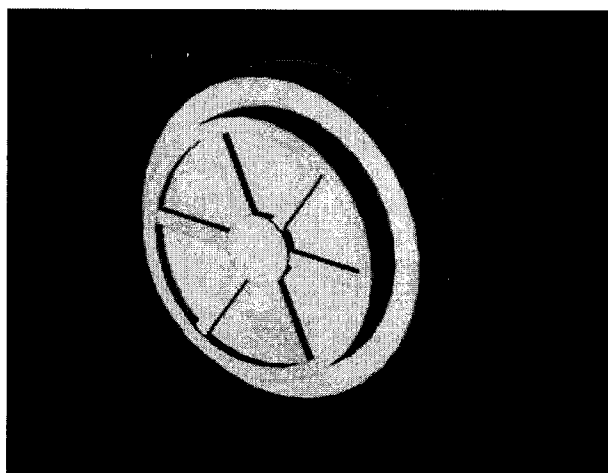


Fig. 17. Wheel solid model imported from SIGGRAPH-3D (Siemens-Nixdorf).



Fig. 18. B-Rep model imported from Unigraphics (EDS).

the *linear accuracy*. A typical value could be 10^{-6} mm which is highly accurate for many mechanical design applications. A similar value can be specified for *angular accuracy*, *parametric accuracy*, and so on. The discussion here is limited to linear accuracy.

If the sending system uses a higher linear accuracy (more precise data) than the receiving system, distinct geometric points will be detected to coincide in the receiving system. This might result in a change in the topology (which might cause further inconsistencies) or the geometry. If the sending system uses a lower linear accuracy (less precise data), the receiving system might complain that the topology is not correct or the geometry and the topology are inconsistent.

To prevent or at least minimize these kinds of accuracy problems it should be possible to adjust the accuracy in the receiving system to the accuracy values of the data to be imported. For example, if the sending system uses a different accuracy for the model generation process, say a linear accuracy of 10^{-2} mm, then the receiving system should adjust its internal algorithms to the same accuracy.

Experience with HP PE/SolidDesigner has shown that this kind of adjustable accuracy helps regenerate CAD models that were generated in different systems with different accuracies. Also, for data models composed of components with different accuracies, the components can be brought together on the assembly level to form a complete product model.

In the STEP implementation of AP214 an adjustable linear accuracy value is conveyed in the STEP file to tell the receiving system the appropriate accuracy value for post-processing.

User Features

The user can select via the HP PE/SolidDesigner graphical user interface the objects (e.g., several B-Rep bodies) to put into a STEP file. For example, the user decides whether to send the data in a B-Rep solid model or a surface model representation. The user can choose some configuration parameters that help tailor the model data set for best communication to a specific target application. However, all data must comply with the STEP standard.

When importing (postprocessing) a STEP file the user can define some parameters that ease the processing of data. For example, the user might set the accuracy value before

importing a data set that was designed with a specified accuracy, or might choose to convert the imported data to a different representation.

STEP Model Exchange Trials

Various STEP file exchanges have been performed within the last 12 months, not always with satisfying results. This has resulted in more development work by the exchange partners. This process of harmonizing the STEP preprocessors and postprocessors of different CAD vendors is considered to be of vital importance for the acceptance of the STEP standard and its application protocols. Within the ProSTEP project this process has worked particularly well. Other work has been done with, for example, AP203 implementors together with PDES Inc.

At this time, solid model data exchange can be said to be working very well, especially compared with what was possible with existing standards. STEP-based surface model exchange has also reached a level that was not possible with existing standards like IGES or VDA-FS, especially with respect to topological coherence, which is easily conveyed with STEP between many CAD systems. Of course, the wide variety of surface models, with the resulting accuracy and connectivity problems, will need to be addressed by the different CAD system vendors to optimize data transfer via STEP. In the meantime, STEP file exchange has matured to the point where STEP products are offered by various CAD vendors and system integrators.

Within the ProSTEP project one of the broadest ranges of STEP-based data exchange trials have been performed between HP PE/SolidDesigner and other CAD systems (see Fig. 14). Solid model industrial part data has been exchanged, for example, with CATIA (CAP Debis and Dassault/IBM), Unigraphics II (EDS), SIGRAPH Design and STEP Viewer (Siemens-Nixdorf), and others. Some of the successful results are shown in Figs. 15, 16, 17, and 18. Surface model industrial part data has been exchanged with CATIA, EUCLID, SYRKO (Mercedes-Benz corporate design system), and others. Some of the successful results are shown in Figs. 19 and 20.

Next Steps

Future releases of the STEP standard covering product data categories such as materials, tolerances, form features, manufacturing process data, and others are expected in the next few months. The expected release of AP202, associative drafting, will allow documentation of the product data in engineering drawings. Work is ongoing towards the parameterization of product features, which needs further development in the STEP standard.

The expected finalization of AP214 will make it possible to convey the product data categories in STEP files and will help to reduce design and manufacturing development cycles for simple as well as complex products. This process will be supported by further extensive use of data communication networks in the various countries. The migration from existing standards is aided by several product offerings of IGES-to-STEP and VDA-FS-to-STEP data converters.

The STEP implementation technology based on the STEP Standard Data Access Interface will be broadened and used

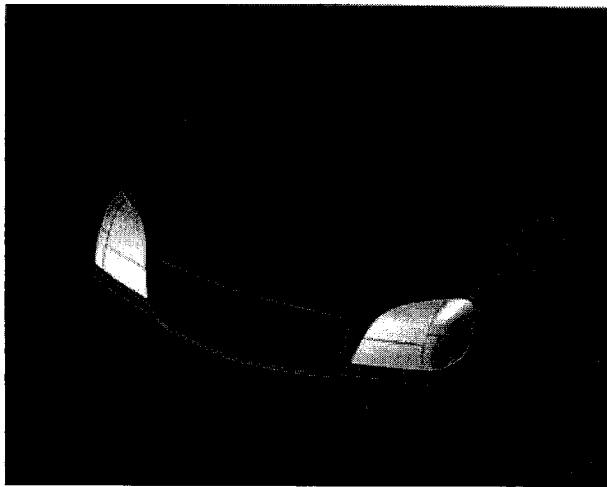


Fig. 19. Surface model imported from SYRKO (Mercedes-Benz corporate design system).

in database access implementations to allow concurrent access by product design and manufacturing development.

However, for industrial use, the database technology and the STEP data access technology need to be extended and integrated. This process is expected to take several years.

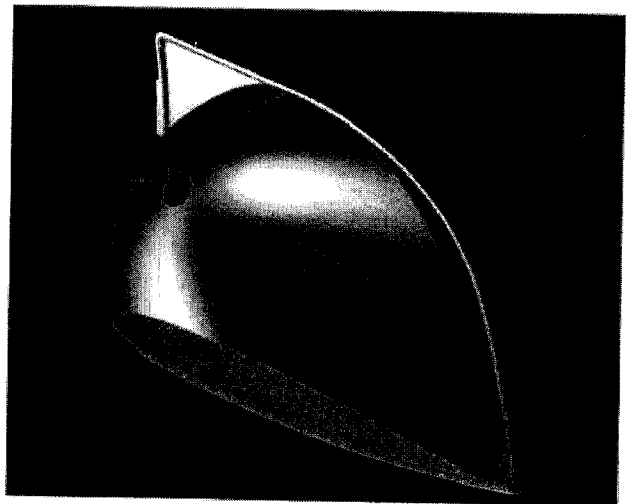


Fig. 20. Headlight reflector surface model imported from EUCLID (Matra Datavision).